Analyse des protocoles cryptographiques: des modèles symboliques aux modèles calculatoires

Véronique Cortier

▶ **To cite this version:**

Véronique Cortier. Analyse des protocoles cryptographiques: des modèles symboliques aux modèles calculatoires. Informatique [cs]. Institut National Polytechnique de Lorraine - INPL, 2009. tel-00578816

HAL Id: tel-00578816
https://theses.hal.science/tel-00578816

Submitted on 22 Mar 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Nancy-Université INPL

Institut National
Polytechnique de Lorraine

Département de formation doctorale en informatique

École doctorale IAEM Lorraine

Analyse des protocoles cryptographiques : des modèles symboliques aux modèles calculatoires

MÉMOIRE

pour l'obtention de

l'Habilitation à Diriger les Recherches

Défendue et soutenue publiquement le 18 novembre 2009

par

Véronique Cortier

Composition du jury

Rapporteurs :	Serge	ABITEBOUL
	Yassine	LAKHNECH
	Roberto	SEGALA
Examinateurs :	Jean	GOUBAULT-LARRECQ
	Joshua	GUTTMAN
	Jean-Yves	MARION
	Michael	RUSINOWITCH
	Jacques	STERN
Invité :	Hubert	COMON-LUNDH

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

À Frédéric,
Ainsi qu'à Margot et Colin.

Remerciements

Je tiens tout d'abord à remercier les membres du jury.

Jacques Stern m'a fait l'honneur de présider le jury, et je l'en remercie vivement.

J'exprime toute ma gratitude à Joshua Guttman pour avoir eu la gentillesse de venir depuis les États-Unis et pour s'être livré de très bonne grâce au « jeu » de l'habilitation, version française. Sa patience et sa réactivité lors de tous les messages que nous avons pu échanger, scientifiques ou non, sont particulièrement appréciables.

Un grand merci à Jean Goubault-Larrecq pour avoir accepté de faire partie de mon jury et pour ses questions, toujours profondes, enrichissantes et nombreuses ! Merci beaucoup à Jean-Yves Marion pour avoir bien voulu porter un regard « extérieur » sur mes travaux ainsi que pour son aide précieuse dans le processus administratif de l'habilitation.

Je remercie chaleureusement Roberto Segala qui a bien voulu s'atteler à la lourde tâche d'écrire un rapport sur ces travaux, tâche d'autant plus difficile que le mémoire est en français, et qui a réussi à intercaler un voyage à Nancy dans un emploi du temps déjà lourdement chargé. Merci également à Serge Abiteboul d'avoir contribué par son rapport à donner un éclairage différent sur mes travaux. Le troisième rapporteur est Yassine Lakhnech, que je remercie non seulement pour sa lecture du mémoire mais également pour toutes les idées, suggestions et conseils qu'il a pu formuler au cours de nos (malheureusement) rares mais régulières conversations tout au long de ces dernières années.

Je tiens à exprimer toute ma gratitude à Michael Rusinowitch pour m'avoir accueillie dans son équipe à mon arrivée au Loria. J'apprécie particulièrement la très grande liberté que Michael m'a toujours accordée tout en étant disponible pour de judicieux conseils. Je le remercie également pour sa très grande patience lors de mes (nombreux !) bavardages et élucubrations scientifiques.

Bien sûr, je tiens à exprimer encore une fois ma profonde reconnaissance à l'« invité » de mon jury, Hubert Comon-Lundh, pour m'avoir enseigné la recherche et m'avoir accompagnée et soutenue depuis que nous nous connaissons. Même si les occasions se sont faites beaucoup plus rares depuis la fin de la thèse, c'est toujours un immense plaisir et beaucoup d'enrichissement de travailler avec lui.

Je souhaite également à remercier toute l'équipe Cassis (Abdessamad, Christophe, Laurent, Mathieu et tous les autres) pour leur disponibilité et gentillesse, et pour toutes les discussions, scientifiques ou non, qui font de cette équipe un cadre de travail idéal. Un merci particulier à Emmanuelle Deschamps pour faciliter grandement les tâches administratives et pour sa patience face à des dossiers incomplets, des changements de dernières minutes ou des montages compliqués. Je remercie également tous les membres (et anciens membres) du LORIA pour faire de ce laboratoire un environnement agréable. Merci en particulier à Claude, Emmanuel, Guillaume, Hazel, Hélène, Johanne, Olivier, Paul, Pierre-Étienne, Pierrick, Sylvain L., Sylvain P., Xavier et tous ceux que j'oublie.

Un grand merci également à tous mes co-auteurs, pour leurs échanges scientifiques bien sûr mais également pour tous les bavardages, pauses-café, repas et même jeux (SET, fléchettes et nombreux autres) qui motivent nos collaborations au moins autant que les problèmes de recherche en eux-mêmes. Merci en particulier à Bogdan, Graham, Martin, Mathieu, Pascal, Ralf, Stéphanie et Steve. Merci également aux nombreux membres et anciens membres du LSV pour les échanges que nous continuons à entretenir. Je n'oublie pas les doctorants et stagiaires que j'ai eu la chance et le plaisir d'encadrer et qui ont tous contribué à ce mémoire. Je remercie en particulier Eugen pour la patience, la persévérance et l'inventivité qu'il a démontré tout au long de notre collaboration. Plusieurs chapitres de ce mémoire lui doivent beaucoup et je souhaite à tout le monde de commencer par un encadrement de thèse aussi aisé. Merci également à Heinrich qui m'a permis d'approfondir certains aspects inexplorés de la modélisation des protocoles cryptographiques. Et merci à Stefan et Mathilde pour toutes les bonnes idées dont ils ont déjà fait preuve et pour les résultats qui ne seront manquer de venir.

Enfin, je voudrais finir par un énorme merci à Frédéric pour toutes ses suggestions, son soutien quotidien (qui passe par sa patience à me laisser discuter de mes moindres interrogations dans tous leurs détails) et ses nombreux encouragements, sans lesquels je n'avancerai pas autant en recherche. Merci en particulier de m'avoir convaincue de m'atteler à la rédaction de ce mémoire et d'avoir bien voulu encore me rassurer en le relisant. Et surtout, merci de me supporter quotidiennement et aussi d'assumer avec brio mes nombreux déplacements, c'est sans aucun doute ce qui fait qu'une forme de « parité » est possible en recherche (au moins dans mon cas) ! Je conclus par une pensée à Margot et Colin qui, pour paraphraser Olivier, m'ont persuadée que ce document ne sera jamais ma meilleure production de ces dernières années.

Table des matières

CHAPITRE 1

Introduction

Préambule. Ce mémoire d'habilitation résume une sélection de mes travaux de recherche depuis ma thèse soutenue en mars 2003. Pour (je l'espère) faciliter la lecture de ce mémoire, j'ai indiqué en italique les références bibliographiques dont je suis co-auteur (comme *[AC06]*). Les autres références sont indiquées en caractère droit (comme [AR00]). D'autre part, pour alléger la présentation, certains résultats sont énoncés sous une forme affaiblie et beaucoup sont décrits de manière informelle. Dans un souci de cohérence de l'ensemble du document, certains résultats comme *[CKW07a]* (sur un modèle cryptographique pour les propriétés de sécurité des protocoles de signature de contrat) ou *[CGLN06]* (sur la combinatoire des ensembles de points finis et répétés, en dimension deux) ne seront pas présentés ici.

1.1 Contexte : vérification de programmes

Il est bien connu que les logiciels comportent de nombreuses erreurs (bugs). Si l'on peut à la rigueur accepter le blocage intempestif de son ordinateur personnel, cela n'est plus acceptable lorsque les logiciels sont utilisés sur des applications critiques telles qu'une voiture, un pacemaker ou une fusée.

Une première approche pour détecter les erreurs dans les logiciels consiste bien sûr à les tester. Cette approche présente cependant l'inconvénient que seul un nombre fini de comportements potentiels sont explorés. Pour des programmes critiques, l'approche retenue à l'heure actuelle est de faire la *preuve* mathématique qu'absolument aucun comportement ne peut entraîner d'erreurs. C'est ce qu'on appelle la vérification de logiciels à l'aide de méthodes formelles. L'ensemble de mes travaux portent sur la vérification d'un type particulier de programmes : les protocoles de sécurité.

Ces protocoles consistent en des règles d'échange entre les points d'un réseau, ils permettent de sécuriser les communications. Ils sont utilisés par exemple dans les distributeurs de billets, les abonnements aux chaînes de télévision payantes, la téléphonie mobile, le commerce électronique. Ils ont pour objectifs de garantir le secret d'une donnée, d'authentifier un des participants, de garantir l'anonymat ou la non-répudiation, etc.

Ces programmes sont exécutés sur des réseaux ouverts facilement accessibles (comme internet), aussi leur analyse nécessite de prendre en compte les attaques arbitraires dont ils peuvent faire l'objet. Il est donc usuel de supposer que les protocoles sont exécutés en présence d'un attaquant qui contrôle les communications du réseau. La présence de cet attaquant amène

des problématiques propres à l'analyse des protocoles, c'est pourquoi un domaine de recherche spécifique aux protocoles cryptographiques s'est développé dans la communauté internationale de la vérification depuis une vingtaine d'années environ.

1.2 Protocoles cryptographiques

Particularités. Les protocoles cryptographiques ont comme première particularité d'avoir recours à des primitives cryptographiques comme le chiffrement, la signature électronique, les fonctions de hachage ou la génération de nombres aléatoires, appelés *nonces*. Représenter les messages échangés par des atomes conduit à une modélisation très grossière qui ne permet pas de refléter quelles parties d'un message sont accessibles et modifiables par les agents. Les modélisations symboliques classiques ont donc recours à une algèbre de termes. Nous discuterons des différents choix possibles au chapitre 2.

D'autre part, il ne s'agit pas seulement de vérifier qu'un protocole est correctement conçu au sens où l'exécution du protocole ne doit pas conduire à des états du système indésirables pour les participants. Il faut s'assurer qu'un protocole réalise ses objectifs même placé dans un environnement arbitraire, qui cherche à faire échouer le protocole. L'environnement est en général appelé *intrus*, *attaquant* ou *adversaire*. Il contrôle les communications et peut donc écouter, bloquer et envoyer des messages sur le réseau. La deuxième particularité des protocoles cryptographiques est donc la prise en compte d'un environnement hostile et arbitraire dont les actions ne sont pas connues à l'avance. En particulier, le système de transitions sous-jacent à l'exécution d'un protocole est toujours à branchement infini.

Définir formellement les propriétés attendues d'un protocole est souvent une étape difficile. Certaines propriétés font désormais consensus comme la confidentialité d'une donnée ou l'authentification des participants. Nous verrons cependant que leur formalisation exacte admet de nombreuses variantes. Ainsi, le secret peut être défini à l'aide d'une propriété d'accessibilité ou au contraire à l'aide d'équivalence observationelle. Les deux définitions sont pertinentes mais conduisent à des notions différentes dont il est intéressant d'étudier les relations. D'autres propriétés comme l'anonymat [HKT94, SMA95, SS96, SH02], l'équité d'un protocole de signature de contrat [GJM99, SM00, KR02, KK05] ou la résistance à la coercition et la vérifiabilité pour les protocoles de vote [DKR09b] sont encore en cours de formalisation.

Un exemple de protocole. Pour illustrer ce mémoire, nous considérerons à plusieurs reprises des variantes du protocole "Wide Mouthed Frog" [BAN89, CJ97] qui permet à deux agents (notés A et B) d'échanger une clef de sessions K_{ab} par l'intermédiaire d'un serveur (noté S).

$$\begin{aligned} A &\to S: \quad A, \{B, K_{ab}\}_{K_{as}} \\ S &\to B: \quad \{A, K_{ab}\}_{K_{bs}} \end{aligned}$$

Chaque agent X possède une clef symétrique long terme K_{xs} avec le serveur. L'agent A souhaite transmettre la clef K_{ab} à l'agent B. À la première étape, l'agent A envoie la clef K_{ab} ainsi que l'identité de B, le tout chiffré par la clef long terme K_{as}. L'agent A indique également son identité à l'extérieur du message. Le serveur déchiffre le message et transmet la clef K_{ab} ainsi que l'identité de A, le tout chiffré par la clef long terme K_{bs}, partagée avec B.

Il est important que l'identité de B soit incluse à l'intérieur de message chiffré à la première

étape. En effet, considérons une variante du protocole, où l'identité de B est en clair.

$$\begin{aligned} A &\rightarrow S : \quad A, B, \{K_{ab}\}_{K_{as}} \\ S &\rightarrow B : \quad \{A, K_{ab}\}_{K_{bs}} \end{aligned}$$

Un agent malhonnête C, possédant une clef long terme K_{cs} partagée avec le serveur, peut alors aisément obtenir toutes les clefs de sessions qu'il souhaite en interceptant le message envoyé par A et en remplaçant l'identité de B par C. L'attaque est schématiquement représentée ci-dessous.

$$\begin{aligned} A &\rightarrow C(S) : \quad A, B, \{K_{ab}\}_{K_{as}} \\ C &\rightarrow S : \quad A, C, \{K_{ab}\}_{K_{as}} \\ S &\rightarrow C : \quad \{A, K_{ab}\}_{K_{cs}} \end{aligned}$$

Remarquons que le protocole initial souffre d'une attaque par *rejeu* : si l'intrus est capable de connaître une ancienne clef K'_{ab}, il peut la faire accepter à B en "rejouant" le deuxième message du protocole :

$$C(S) \rightarrow B : \quad \{A, K'_{ab}\}_{K_{bs}}$$

L'agent B va alors utiliser la clef K'_{ab} (connue de C) pour chiffrer ses messages à l'attention de A. Pour éviter ce type d'attaque, le protocole présenté dans [BAN89, CJ97] comporte des "horodatages" (timestamps) qui permette de détecter le rejeu de données anciennes. Dans les chapitres suivants, nous verrons comment ce protocole peut être représenté dans les différents modèles étudiés et comment nos techniques permettent de retrouver les attaques évoquées.

1.3 Travaux existants

Depuis les années 80, deux approches ont été développées pour l'analyse des protocoles de sécurité. L'une de ces approches repose sur un modèle calculatoire qui prend en compte les probabilités et la complexité algorithmique. Cette approche permet de définir une notion très forte de sécurité, garantissant contre n'importe quelles attaques probabilistes et polynomiales. L'autre approche repose sur une modélisation symbolique des exécutions du protocole, où les primitives cryptographiques sont traitées comme des boîtes noires. Depuis le travail précurseur de Dolev et Yao, différents modèles symboliques ont été proposés, nous mentionnons les principaux modèles à la partie 1.3.1. Les modèles symboliques présentent l'avantage de permettre des preuves de sécurité beaucoup plus simples que dans les modèles calculatoires et souvent automatiques. Nous en décrivons les principales techniques à la partie 1.3.2.

Cependant, le niveau de garanties offert par les preuves de sécurité dans les modèles symboliques n'est pas clair. Depuis plus de vingt ans, les deux approches ont évolué de manière plutôt indépendante. Des travaux de recherche récents cherchent à développer des techniques pour combiner le meilleur des deux approches. Nous évoquons les principales directions de ces travaux à la partie 1.3.3.

1.3.1 Modèles symboliques

De nombreux modèles symboliques ont été proposés pour les protocoles cryptographiques. Un seul modèle unifié permettrait de mieux comparer les résultats et leurs hypothèses mais le

domaine de la vérification symbolique des protocoles ne semble pas avoir encore atteint une maturité suffisante pour définir un modèle unique et consensuel. La co-existence de nombreux modèles est due au fait que la modélisation symbolique doit répondre à deux buts antagonistes : d'une part permettre une modélisation la plus fine et la plus expressive possible pour refléter au mieux les protocoles et d'autre part conserver une relative simplicité pour permettre la conception de procédures de décision, souvent automatiques.

Modèle de D. Dolev et A.C. Yao. L'un des premiers modèles est celui développé par D. Dolev *et al.* [DY81, DEK83]. Les protocoles sont décrits par des règles de réécriture de mots ; un mot s est secret s'il n'est pas accessible par réécriture. M. Merritt *et al.* [RDM82, Mer83] ont développé pendant la même période un modèle où les messages sont également représentés sous forme de mots. Ces modèles ne sont cependant pas assez expressifs pour représenter de nombreuses primitives comme la concaténation de messages ou la création de clefs.

Systèmes de contraintes. Les systèmes de contraintes ont été introduits par Jon Millen [MS01, MS03] puis repris par Hubert Comon-Lundh [CL04]. Ils sont particulièrement bien adaptés pour représenter l'exécutions des protocoles pour un nombre borné de sessions. Nous les présenterons au chapitre 3 et proposerons une procédure de décision.

Règles de réécriture. Plusieurs modèles représentent les règles des protocoles et le pouvoir de l'intrus par des règles de réécriture sur des termes. Citons principalement le MultiSet Rewriting (MSR) développé par J. Mitchell *et al.* [CDL$^+$99, BCJS02], les règles de réécriture utilisées par M. Rusinowitch et M. Turuani dans l'outil Casrul [RT01] et le modèle basé sur la logique linéaire de K. Compton et S. Dexter [CD99]. La principale limitation de cette modélisation est de ne pas toujours exprimer les nonces de manière fidèle et de permettre la répétition de règles déjà jouées.

Clauses de Horn. Une variante de la modélisation sous forme de réécriture est la modélisation sous forme de clauses de Horn [Wei99, Bla01, Bla04, VSS05], *[CLC03a]*. Ces deux formes de modélisation sont très proches et les difficultés rencontrées sont également similaires. La modélisation sous forme de clauses de Horn présente cependant l'avantage de permettre l'utilisation des techniques déjà développées sur les clauses comme les stratégies de résolution. Ainsi, l'un des outils les plus utilisés à l'heure actuelle pour la vérification des protocoles pour un nombre non borné de sessions est l'outil ProVerif développé par Bruno Blanchet [Bla01, Bla05, BAF08]. Dans cet outil, les protocoles et propriétés de sécurité sont modélisés sous forme de clauses de Horn et l'outil intègre des stratégies de résolution dédiées. Nous présenterons ce modèle ainsi que de nouveaux résultats de décidabilité au chapitre 4. Nous verrons également que les clauses de Horn se sont révélées bien adaptées pour modéliser un nouveau type de systèmes que sont les interfaces de programmation dédiées aux modules matériels de sécurité.

Strand spaces. Le modèle des strand spaces [THG99, GT01] est un modèle de traces où les transitions entre messages sont exprimées explicitement dans chaque historique. Cela permet en particulier de formaliser la représentation graphique « intuitive » des déroulements de protocoles. C'est à l'aide de ce modèle que Joshua Guttman a développé plusieurs résultats de *composition* des protocoles [GT00, Gut04, Gut09], que nous évoquerons à la partie 5.

Algèbres de processus. Un autre type de modélisation est la représentation des protocoles par des processus. Cela correspond à une modélisation plus proche de l'implémentation des protocoles. En effet, chaque rôle est représenté par un processus indépendant des processus représentant les autres rôles. Ces algèbres de processus (comme CSP [Sch97], le spi-calcul [AG97] ou le pi-calcul appliqué [AF01]) permettent en général l'émission et la réception de messages, le filtrage de messages (lors de la réception) ainsi que la restriction, la composition et la réplication de processus. Les algèbres de processus permettent donc souvent une modélisation (symbolique) fidèle des exécutions possibles en permettant à la fois la création de nonces frais ainsi que l'exécution d'un nombre arbitraire de sessions.

La principale différence entre les différents modèles utilisant des algèbres de processus est la modélisation de la propriété de sécurité. Beaucoup de modèles [AL00, ALV02, Sch96a, Sch97, Bor01] ramènent les propriétés de sécurité à une propriété d'accessibilité de la forme : $P \xrightarrow{*} err$. D'autres [AG98, BDNP99, AG97, AF01] définissent une notion d'*équivalence observationelle* qui permet de modéliser de plus grande famille de propriétés comme les propriétés de type anonymat ou résistance à la coercition dans les protocoles de votes [KR05, DKR09b]. Nous décrirons le pi calcul appliqué au chapitre 7.

1.3.2 Procédures de décisions

Les modèles symboliques évoqués dans la partie précédente sont à la fois suffisament riches pour capturer les plupart des attaques ne reposant pas sur la cryptographie et également suffisament simples pour permettre l'automatisation des preuves de sécurité et de la recherche d'attaques.

1.3.2.1 Décidabilité

De nombreux résultats de décidabilité ont été développés dans le contexte de la vérification des protocoles cryptographiques. Tout d'abord, plusieurs résultats sont négatifs et permettent de cerner la difficulté du problème. J. Mitchell *et al.* [DLMS99] ont montré que le secret est indécidable pour des protocoles avec nonces, avec des messages de taille bornée et un nombre de sessions non borné. R. Amadio et W. Charatonik [AC02] ont raffiné ce résultat à des protocoles ne possédant qu'une seule primitive cryptographique : le chiffrement. Il est donc nécessaire de mettre au point des restrictions raisonnables qui permettent d'obtenir des procédures de décision.

Nombre borné de sessions. La première restriction considérée est de restreindre le nombre de sessions à analyser. Dans le cadre des protocoles avec chiffrement (symétrique et asymétrique) et concaténation, M. Rusinowitch et M. Turuani [RT01, RT03] ont montré que le secret est un problème co-NP-complet. Dans le contexte des algèbres de processus, R. Amadio *et al.* [AL00, ALV02] ont monté un résultat similaire mais sans modéliser les clefs composées. Des résultats de décidabilité très proches ont également été établis, toujours pour des processus sans réplication par [Bor01, FA01, MS01].

Nombre non borné de sessions. L'analyse des protocoles pour un nombre borné de sessions est très bien adaptée pour découvrir des attaques. Cependant, lorsqu'aucune attaque n'a été trouvée, il n'est pas possible de conclure sur la sécurité du protocole, d'autant que les outils ne permettent d'explorer qu'un nombre très limité (en général deux ou trois) de sessions. La mise

au point de classes décidables pour un nombre non borné de sessions nécessite cependant de limiter de manière importante les classes de protocoles considérées. Si la taille des messages est bornée, J. Mitchell *et al.* [DLMS99] ont montré que le secret est décidable pour des protocoles sans nonce. S. Fröschle a raffiné le résultat en autorisant la création de nonces pour les agents malhonnêtes [Fro07].

Au cours de ma thèse, nous avons montré que le secret est décidable pour les protocoles (sans nonce frais) ne permettant qu'au plus une copie par transition *[CCM01, CLC03a]*. K. Verma *et al.* ont montré que la complexité de la décision du secret est DEXPTIME-complet [SV05, SV08]. B. Blanchet et . Podelski ont montré que l'ajout d'étiquettes permet d'obtenir la décidabilité pour les protocoles sans nonce frais [BP03, BP05c]. R. Ramanujam et S. Suresh permettent le traitement d'un nombre arbitraire de nonces en mettant en œuvre un système d'étiquetage nettement plus fort et pour des protocoles sans copie de messages arbitraires [RS03]. La question de la décidabilité pour des protocoles étiquetés au sens défini par Bruno Blanchet [BP03] et pour des protocoles avec nonces reste un problème ouvert. Myrto Arapinis et Marie Duflot ont obtenu un premier résultat en montrant que l'étiquetage des protocoles (avec nonces) permet de borner la taille des messages nécessaires à une attaque [AD07]. Ce résultat ne permet cependant pas de conclure à la décidabilité car les nonces restent en nombre non borné.

Y. Chevalier *et al.* ont montré [CKR$^+$03b] qu'il était possible de combiner un nombre borné de sessions à un nombre non borné de sessions en ajoutant des règles d'oracles au système de déduction de l'intrus.

Théories équationelles. Tous les résultats de décidabilité évoqués au paragraphe précédent sont obtenus pour des primitives relativement simples à modéliser que sont le chiffrement et la concaténation. D'autres primitives comme le « ou exclusif », les groupes abéliens ou les signatures en aveugle, demandent l'introduction de *théories équationelles* pour refléter les propriétés algébriques de ces opérateurs. Adapter les techniques de décision à ces primitives plus complexes est souvent difficile. De nombreuses nouvelles classes de décidabilité ont cependant été obtenues. Nous en mentionnons quelques unes ici.

Pour le « ou exclusif », Hubert Comon-Lundh et Vitaly Shmatikov [CLS03] ainsi que Yannick Chevalier *et al.* [CKRT03, CKRT05] ont montré que le secret était décidable pour un nombre borné de sessions. Au cours de ma thèse, nous avons montré *[CLC03a]* que le secret est également décidable pour un nombre non borné de sessions, pour des protocoles avec une seule copie par transition. Pour les groupes abéliens, Jon Millen et Vitaly Shmatikov [MS03, Shm04] ont proposé une classe décidable pour un nombre borné de sessions. Kumar Verma [Ver03] a obtenu un résultat de décidabilité pour un nombre non borné de sessions, pour des protocoles pouvant être représentés par des automates bi-directionels pour des théories avec symboles associatifs et commutatifs.

Pour traiter des protocoles comme celui de Diffie-Hellman [DH76], il est nécessaire de considérer également certaines propriétés de l'exponentiation modulaire. Dans ce cadre et pour un nombre borné de sessions, Michele Boreale et Maria Buscemi [BB03] d'une part et Yannick Chevalier *et al.* [CKR$^+$03a] d'autre part en obtenu des résultats de décidabilité en bornant le nombre de sessions et en restreignant les exposants considérés dans l'exponentiation modulaire. Jean Goubault-Larrecq *et al.* [GLRV04] a développé une implémentation permettant d'analyser en pratique certains protocoles faisant usage de l'exponentiation modulaire.

Nous proposons dans l'article *[CDL06]* un état de l'art plus complet des résultats de

décidabilité obtenus dans le contexte de la vérification des protocoles cryptographiques.

1.3.2.2 Outils

Plutôt que d'identifier des classes décidables, une alternative consiste à développer des (semi) procédures de décision qui sont correctes (une preuve de sécurité obtenue par l'outil garantit la sécurité du protocole) mais qui ne terminent pas toujours ou peuvent échouer. C'est ce type d'approche qui est utilisé dans les outils développés pour l'analyse des protocoles pour un nombre non borné de sessions. Nous en citons ici quelques-uns. Ainsi, nous avons déjà mentionné l'outil ProVerif [Bla01, Bla05] basé sur des stratégies de résolution pour les clauses de Horn, correctes mais incomplètes (possibilité de trouver de fausses attaques) et sans garantie de terminaison. Cet outil est très performant en pratique et a été utilisé pour analyser de nombreux protocoles (par exemple [ABF04, ABF07, AB05, BC08]). L'outil ProVerif a la particularité de pouvoir traiter aussi bien les propriétés d'accessibilité comme le secret ou l'authentification [Bla02] que les propriétés d'équivalence à l'aide de sur-approximations [Bla04, BAF05, BAF08].

L'outil TA4SP [BHKO04, BHK07] calcule une sur-approximation de l'ensemble des messages accessibles à l'intrus à l'aide d'automates d'arbre. Si le secret n'appartient pas à la sur-approximation, le protocole est sûr.

L'outil Scyther [Cre08a, Cre08b, CLN09] manipule symboliquement (à l'aide de « patterns ») l'ensemble des traces et peut, assez souvent, conclure à la sécurité d'un protocole pour un nombre borné de sessions. L'intérêt de cet outil est de permettre un calcul *exact* pour un nombre borné (et fixé par l'utilisateur) de l'outil. Ainsi, même si lorsque l'outil ne permet pas de conclure dans le cadre d'un nombre non borné de sessions, il permet de savoir s'il existe ou non une attaque pour le nombre de sessions choisi.

Une autre famille d'outils se consacre à l'analyse des protocoles pour un nombre borné de sessions. La difficulté réside alors dans la construction d'un outil le plus efficace possible pour qu'il soit capable de traiter des exemples relativement importants de protocoles, pour souvent deux à trois sessions. Parmi ces outils, citons par exemple Casper/FDR [LR97, Low97, RSG+00] qui a permis de détecter la célèbre « man-in-the-middle » attaque [Low96] sur le protocole de Needham-Schroeder à clef publique [NS78]. Les outils Atse [Tur06], OFMC [BMV05] et Sat-MC [AC05] sont d'autres exemples d'outils très utilisés.

1.3.3 Lien avec la cryptographie

Tous les travaux précédents se placent dans le cadre de modèles symboliques, où les primitives cryptographiques sont représentées par des symboles fonctionnels, éventuellement munis d'une théorie équationelle. Ces modèles sont très différents de ceux utilisés en cryptographie, pour concevoir (par exemple) des algorithmes de chiffrement. La notion de « sécurité » dans la cryptologie contemporaine est basée sur la théorie de la complexité. Les messages sont représentés par des suites finies de bits et les fonctions de chiffrement comme des algorithmes qui opèrent sur ces suites. La question est alors de savoir si on peut construire un adversaire (une machine de Turing) qui est capable, par exemple, d'apprendre une information confidentielle dans un temps raisonnable (polynomial) et avec une probabilité non négligeable. Cette notion de sécurité semble être mieux adaptée pour identifier toutes les attaques possibles dans la réalité mais, en contrepartie, les (lourdes) preuves de sécurité sont effectuées à la main et semblent difficilement automatisables.

Les modèles symboliques et les modèles cryptographiques (ou calculatoires) sont à première vue difficilement conciliables. En effet, il est déjà difficile d'obtenir des procédures automatiques de vérification dans un cadre idéalisé, il pourrait sembler impossible d'intégrer les notions de sécurité utilisées par les cryptographes. Cependant, depuis le début des années 2000, une nouvelle ligne de recherche esquisse un rapprochement entre ces deux approches. Nous avons établi un état de l'art *[CKW09]* le plus complet possible sur les travaux qui visent à utiliser les modèles symboliques pour obtenir des garanties dans les modèles cryptographiques. Nous résumons ici les principales directions de recherche. Ainsi, Martín Abadi et Philipp Rogaway ont ouvert la voie en montrant qu'il était possible d'abstraire l'indistinguabilité d'une séquence de message par une notion symbolique d'équivalence de séquences de termes, définie à l'aide de « patrons » (appelés « patterns » en anglais) [AR00]. Ces travaux ont été suivis de nombreuses extensions à d'autres primitives cryptographiques [Her03, Her05, LC04, GvR08, BLMW07, Maz07, KM09]. Dans le cas actif, Michael Backes *et al.* ont proposé une bibliothèque symbolique permettant d'abstraire les opérations cryptographiques [SBB+06, BP05b]. Leur bibliothèque permet de traiter de nombreuses primitives (signatures et chiffrement asymétrique [BPW03], chiffrement symétrique [BP04], protocoles à divulgation nulle [BU08], etc.). Peter Laud a également proposé une procédure de décision symbolique pour des preuves cryptographiques de protocoles à chiffrement symétrique [Lau04]. John Mitchell *et al.* ont proposé une logique symbolique (PCL) [DDM+05, DDMW06, RDDM07] qui permet d'obtenir des garanties dans les modèles cryptographiques.

1.4 Contributions

1.4.1 Cheminement

J'ai effectué ma thèse entre 2000 et 2003 au sein du LSV (à l'École Normale Supérieure de Cachan) sous la direction d'Hubert Comon-Lundh. Mon travail de thèse portait sur la vérification automatique des protocoles cryptographiques. J'ai ensuite été recrutée en 2003 comme chercheur CNRS au sein de l'équipe Cassis du LORIA, dirigée par Michael Rusinowitch. À l'issue de ma thèse *[Cor03b]* en 2003, l'état de l'art autour de la vérification automatique des protocoles cryptographiques permettait de bien identifier les différentes classes de décidabilité et indécidabilité. Ainsi, il était établi que le secret est indécidable en général [DLMS99], NP-complet pour un nombre borné de sessions [RT01] et certaines classes décidables avaient été identifiées pour un nombre non borné de sessions [DLMS99, BP03, RS03] *[CCM01, CLC03a]*.

Théories équationelles. La première question naturelle est l'extension de ces résultats à des primitives cryptographiques plus complexes comme les groupes abéliens, l'exponentiation modulaire, les signatures en aveugles ou le chiffrement CBC pour citer quelques exemples. Ces primitives demandent au minimum de définir un système de déduction plus complexe pour l'intrus et il est souvent nécessaire d'introduire une théorie équationelle. Des premiers résultats avaient déjà été obtenus pour le « ou exclusif » [CLS03, CKRT03] *[CLC03a]*. Lors du début de la thèse d'Eugen Zalinescu, nous avons identifié *[CRZ05]* un nouveau fragment décidable de clauses de Horn permettant en particulier d'analyser les protocoles pour des primitives telles que le chiffrement en aveugle ou le chiffrement CBC et pour un nombre non borné de sessions. D'autre part, grâce à une collaboration débutée fin 2003 avec Martín Abadi, nous avons démarré une étude assez complète de la décidabilité de l'équivalence statique

pour diverses théories équationelles *[AC04a, AC05, AC06]*. La notion d'équivalence statique permet de définir si un intrus est capable de différencier deux séquences de messages. C'est une notion plus fine que la déduction et au cœur de certaines propriétés comme l'anonymat ou la confidentialité d'un vote. Nous avons poursuivi l'étude de l'équivalence statique, notamment avec Mathilde Arnaud et Stéphanie Delaune pour la combinaison de théories *[ACD07a]* et les théories monoïdales *[CD07a]*, avec Mathieu Baudet pour une procédure plus efficace et implémentée *[BCD09a]* et récemment avec Mouhebeddine Berrima et Narjes Ben Rajeb pour des théories équationelles adaptées aux protocoles de vote *[BBRC09]*.

Nouvelles familles de protocoles. D'autre part, les modèles développés jusqu'en 2003 étaient bien adaptés pour traiter les protocoles « classiques » comme ceux référencés dans la bibliothèque de Clark & Jacob [CJ97]. Ces modèles se sont cependant révélés insuffisants pour les familles de protocoles apparues ensuite comme les protocoles de groupe, les protocoles de signatures de contrat ou les protocoles de vote. À l'occasion d'une collaboration avec Graham Steel, nous nous sommes particulièrement intéressés aux interfaces de programmation (API) dédiées aux modules matériels de sécurité. Ces interfaces permettent de séparer la partie sécurisée d'une machine (le module de sécurité) où il est possible de stocker des données sensibles de la partie non sûre. Nous avons proposé plusieurs modélisations et des nouveaux résultats de décidabilité *[CKS07, CDS07a]* ainsi qu'une nouvelle interface générique *[CS09a]*.

Nous nous sommes également intéressés aux protocoles de vote. Ces protocoles ont plusieurs spécificités. D'une part, les primitives cryptographiques sont assez particulières comme les signatures en aveugle ou le rechiffrement. Adapter les techniques de décidabilité à ces nouvelles primitives est délicat. Nous avons obtenu des premiers résultats *[CRZ05, BBRC09]* mais il s'agit d'un travail toujours en cours car il n'est toujours pas possible de traiter complètement le cas des protocoles de vote pour un intrus actif. D'autre part, ces protocoles ont des buts de sécurité assez particuliers comme la confidentialité des votes ou la résistance à la coercition. Ces propriétés ont été formalisées à l'aide de la notion d'équivalence observationelle [KR05, DKR09b]. Avec Stéphanie Delaune, nous avons proposé *[CD09a]* une procédure de décision pour l'équivalence observationelle en nous ramenant à un résultat de décidabilité mis au point par Mathieu Baudet [Bau05, Bau07].

Résultats de transfert. À mesure que le domaine de la vérification automatique des protocoles cryptographiques gagnait en maturité, il nous a semblé intéressant de développer des résultats de transfert permettant de réutiliser des résultats existants. Ainsi, avec Eugen Zalinescu et Michael Rusinowitch, nous avons identifié *[CRZ06, CRZ07]* sous quelles conditions la notion de secret définie sous forme d'indistinguabilité (dit secret fort) coïncide avec la notion plus usuelle de secret. Avec Stéphanie Delaune et Jérémie Delaitre, nous avons proposé *[CDD07a, CD09c]* d'étiqueter les protocoles ce qui permet d'utiliser un protocole sûr en même temps que d'autres protocoles partageant des clés, sans avoir à refaire la preuve de sécurité pour l'ensemble des protocoles. Enfin, avec Bogdan Warinschi et Eugen Zalinescu, nous avons proposé une transformation *[CWZ07b]* qui permet de passer d'un protocole sûr pour une seule session et sans intrus à un protocole sûr pour un intrus actif et un nombre non borné de sessions.

Lien avec la cryptographie. Dès la fin de la thèse, je me suis demandée quelles sont les garanties offertes par les modèles symboliques : lorsqu'un protocole est prouvé sûr, est-il vrai-

ment sans faille ? En particulier, nous avons vu à la partie 1.3.3 que les modèles symboliques s'affranchissent complètement des primitives cryptographiques réellement utilisées et des hypothèses calculatoires associées.

Lors de ma première visite à Martín Abadi, j'ai eu l'occasion de rencontrer également Bogdan Warinschi (nous partagions le même bureau) qui venait de publier un premier résultat montrant qu'un modèle symbolique, proche des modèles existants, permettait d'obtenir directement des preuves au niveau cryptographique. Ce résultat présentait certaines limitations comme le fait de ne considérer qu'une seule primitive (le chiffrement asymétrique) et de ne pas permettre la transmission de paquets non déchiffrables (comme des « tickets ») par des agents. Nous avons étendu ce résultat à de nouvelles primitives comme les signatures *[CW05]* ou les fonctions de hachage *[CKKW06]*.

Identifier des hypothèses calculatoires permettant d'obtenir des preuves cryptographiques à l'aide des modèles symboliques a comme premier intérêt d'automatiser les preuves dans les modèles calculatoires. Une autre perspective intéressante est de mesurer l'exactitude de la modélisation symbolique. Ainsi, lorsqu'on ajoute des théories équationelles aux modèles symboliques, peut-on être sûr d'avoir considéré suffisamment d'équations pour capturer toutes les attaques cryptographiques possibles ? La question se pose de manière très naturelle dans le cadre de l'équivalence statique qui définit symboliquement la capacité d'un intrus à distinguer deux séquences de messages. Cette notion a son pendant calculatoire, l'indistinguabilité. Avec Mathieu Baudet et Steve Kremer, nous avons examiné *[BCK05, BCK09]* sous quelles conditions l'équivalence statique implique l'indistinguabilité calculatoire pour des primitives cryptographiques arbitraires. Nous avons poursuivi ce travail dans le cas actif avec Hubert Comon-Lundh *[CLC08a]*, cette fois pour des primitives fixes (chiffrement asymétrique et concaténation).

Relier les modèles symboliques aux modèles calculatoires demande souvent de traiter de nouvelles propriétés au niveau symbolique comme l'absence de cycles de clefs ou des notions de secret adaptées à des primitives comme les fonctions de hachage. Cela nous a amené à revisiter et développer *[CZ06, CCLZ10]* la procédure de décision proposée par Hubert Comon-Lundh [CL04] pour les systèmes de contraintes.

État de l'art. Nous avons également rédigé deux articles permettant de faire le point sur les résultats du domaine. Le premier *[CDL06]* fait un tour d'horizon sur les résultats de décidabilité obtenus pour les protocoles cryptographiques, notamment en présence de primitives cryptographiques nécessitant l'introduction de théories équationelles. Le deuxième *[CKW09]* propose un panorama sur les résultats permettant d'obtenir des garanties cryptographiques à l'aide de méthodes symboliques.

1.4.2 Plan

J'ai organisé la présentation de mes contributions selon trois parties. La première partie présente les résultats obtenus dans les modèles symboliques pour les propriétés de type accessibilité comme le secret ou l'authentification. Cette partie regroupe une procédure de décision développée pour les systèmes de contraintes et les résultats de décision qui en découlent (chapitre 3), des techniques développées pour des fragments de clauses de Horn, notamment pour les interfaces de programmation de sécurité (chapitre 4), ainsi que des résultats de transfert pour la sécurité des protocoles (chapitre 5).

La deuxième partie regroupe les résultats obtenus dans les modèles symboliques pour les propriétés de type équivalence pour diverses théories équationelles dans le cas passif (chapitre 6) ainsi que dans le cas actif (chapitre 7).

Enfin, la troisième et dernière partie de ce mémoire présente les résultats permettant d'obtenir des preuves cryptographiques à partir des modèles symboliques, tout d'abord pour les propriétés d'accessibilité (chapitre 8) puis pour les propriétés d'équivalence (chapitre 9).

Avant d'aborder ces trois parties, le chapitre 2 propose une (courte) description des caractéristiques communes aux modèles symboliques pour les protocoles cryptographiques.

1.4.3 Collaborations

Tous ces travaux ont été réalisés en collaboration avec de nombreux chercheurs et n'auraient pas été possibles sans eux. La liste ci-dessous mentionne la plupart de mes co-auteurs depuis mes travaux de thèse. Les établissements mentionnés sont les institutions d'appartenance en mai 2009. Une liste complète des co-auteurs est présentée à la fin de ce mémoire, page 135.

Mathilde Arnaud, *LSV* (France)
Martín Abadi, *Microsoft Research Silicon Valley* (USA)
Mathieu Baudet, *DCSSI* (France)
Narjes Ben Rajeb, *LIP2* (Tunisie)
Mouhebeddine Berrima, *LIP2* (Tunisie)
Hubert Comon-Lundh, *AIST / LSV* (Japon / France)
Jérémie Delaitre (France)
Stéphanie Delaune, *LSV* (France)
Heinrich Hördegen (Allemagne)
Gavin Keighren, *Université d'Edimbourg* (UK)
Steve Kremer, *LSV* (France)
Ralf Küsters, *Université de Trier* (Allemagne)
Pascal Lafourcade, *Verimag* (France)
Michael Rusinowitch, *Loria* (France)
Graham Steel, *LSV* (France)
Bogdan Warinschi, *Université de Bristol* (UK)
Eugen Zalinescu, *Centre de Recherche Commun INRIA-Microsoft* (France)

En particulier, ce travail repose sur deux thèses que j'ai co-encadrées entre 2004 et 2007.

- La thèse d'Eugen Zalinescu sur la *sécurité des protocoles cryptographiques : décidabilité et résultats de transfert*, soutenue le 17 décembre 2007 et co-encadrée avec Michael Rusinowitch. Les travaux d'Eugen sont décrits aux chapitres 3, 4 (partie 4.2), 5 (partie 5.1) et 7 (partie 7.2).
- La thèse d'Heinrich Hördegen sur la *vérification des protocoles cryptographiques : comparaison des modèles symboliques – Étude des protocoles récursifs*, soutenue le 29 novembre 2007 et co-encadrée avec Michael Rusinowitch. Les travaux d'Heinrich sont mentionnés au chapitre 8.

Ce travail repose également sur des résultats développés à l'occasion d'encadrements de stages de recherche. Je tiens en particulier à mentionner Mbarka Mabrouki (stage de Master 2 en 2005), Mathilde Arnaud (stage de 1ère année de l'ENS Cachan en 2006, le travail effectué est présenté partie 6.6 et a donné lieu à la publication *[ACD07a]*), Jérémie Delaitre (stage de Master 2 en 2007, le travail effectué est présenté partie 5.2 et a donné lieu à la

publication *[CDD07a]*), Walid Chaabene (stage de fin d'études de l'École Polytechnique de Tunisie en 2008) ainsi que Mouhebeddine Berrima (stage de recherche dans le cadre d'un projet Franco-Tunisien en 2008, le travail effectué est présenté partie 6.5).

D'autre part, une partie des perspectives décrites dans le dernier chapitre de ce mémoire seront développées dans le cadre de deux thèses débutées à l'automne 2008.

- La thèse de Mathilde Arnaud sur la *vérification de protocoles dans les réseaux sans-fil*, co-encadrée avec Stéphanie Delaune.
- La thèse de Stefan Ciobaca sur *l'étude des propriétés d'équivalence observationelle et l'application aux protocoles de vote*, co-encadrée avec Steve Kremer.

CHAPITRE 2

Préliminaires : algèbre de termes

Tous les modèles symboliques dédiés aux protocoles cryptographiques ont pour point commun de représenter les messages à l'aide de termes. Ainsi, les noms des agents, les clefs et les *nonces* (nombres générés aléatoirement lors de l'exécution du protocole) sont en général représentés par des constantes. Lorsqu'on considère un nombre non borné de sessions, un nombre arbitraire de nonces et de clefs peuvent être générées. Afin de considérer une signature finie, les nonces et les clefs sont alors souvent représentés par des *noms*. Un symbole fonctionnel est associé à chaque primitive cryptographique. La concaténation de deux messages m_1, m_2 est souvent notée $\langle m_1, m_2 \rangle$ ou $\mathsf{pair}(m_1, m_2)$. Le chiffrement d'un message m par une clef k est en général noté $\{m\}_k$ ou $\mathsf{enc}(m, k)$.

2.1 Algèbre de termes

Formellement, noua considérerons donc une signature $\mathcal{F}$, c'est-à-dire un ensemble de symboles fonctionnels donnés avec leur arité. Nous considérerons également un ensemble $\mathcal{X}$ de variables et un ensemble $\mathcal{N}$ de *noms*, disjoints de $\mathcal{F}$. L'algèbre de termes $T(\mathcal{F}, \mathcal{N}, \mathcal{X})$ formée sur $\mathcal{F}$, $\mathcal{N}$ et $\mathcal{X}$ est définie inductivement par

$$T(\mathcal{F}, \mathcal{N}, \mathcal{X}) := \mathcal{X} \mid \mathcal{N} \mid f(t_1, \ldots, t_n) \qquad t_i \in T(\mathcal{F}, \mathcal{X})$$

où $f \in \mathcal{F}$ est d'arité n. Nous pourrons également considérer une algèbre de termes avec *sortes* (ou typage).

Les capacités de l'intrus sont souvent représentées à l'aide d'un système de déduction. Le système de déduction classique (souvent appelé Dolev-Yao) associé à la paire et au chiffrement (symétrique) est constitué des cinq règles décrites à la figure 2.1.

2.1.1 Discussion

Les algèbre de termes utilisées peuvent varier d'un modèle à l'autre.

1. Des primitives peuvent être ajoutées comme les signatures, souvent notées sign ou les fonctions de hachage, souvent notées h.
2. Deux types de chiffrement peuvent être utilisés : le chiffrement symétrique (souvent noté enc) ou asymétrique (souvent noté enca). Deux symboles distincts sont souvent utilisés

$$\frac{x \quad y}{\mathsf{enc}(x,y)} \qquad \frac{\mathsf{enc}(x,y) \quad y}{x}$$

$$\frac{x \quad y}{\mathsf{pair}(x,y)} \qquad \frac{\mathsf{pair}(x,y)}{x} \qquad \frac{\mathsf{pair}(x,y)}{y}$$

FIGURE 2.1 - *Système de déduction Dolev-Yao.*

mais le même symbole peut également représenter les deux types de chiffrement, auquel cas le type de chiffrement est indiqué par le type de la clef.

3. Pour rendre mieux compte de l'implémentation, un symbole ternaire peut être préféré pour modéliser le chiffrement et représenter son caractère probabiliste : le même message m chiffré à deux instants différents par une même clef k ne donnera pas le même chiffré. Le chiffrement de m par k est alors souvent noté $\mathsf{enc}(m,k,r)$ où r représente l'aléa du chiffrement.
4. Une dernière différence importante est l'utilisation possible de destructeurs explicites. Le plus souvent, la possibilité de chiffrer et déchiffrer des messages est modélisée par un système de déduction comme celui de la figure 2.1. Une alternative consiste à introduire un symbole fonctionnel dec pour le déchiffrement, ainsi que l'équation

$$\mathsf{dec}(\mathsf{enc}(x,y),y) = x.$$

Les deux modélisations sont proches sans être équivalentes. Ainsi, Jon Millen [Mil03] a montré que certaines attaques ne seront détectées que lorsque les destructeurs sont explicitement représentés.

L'intérêt des théories équationelles est de permettre plus de souplesse. En effet, le système de déduction sous-jacent en découle immédiatement et n'a pas besoin d'être défini à la main pour chaque primitive. Il est donc possible de considérer une signature arbitraire. D'autre part, le choix d'une modélisation à l'aide de théories équationelles permet de considérer plus de propriétés comme les propriétés d'équivalence, souvent plus fines que les propriétés d'accessibilité.

La modélisation à l'aide de systèmes de déduction est cependant souvent plus facile à manipuler dans les preuves et évite de considérer des termes irréductibles contenant des destructeurs, comme par exemple le terme $\mathsf{dec}(\langle m_1, m_2\rangle, k)$, qui correspondent est en général un message d'erreur (cela dépend bien sûr de l'implémentation des primitives).

Nous serons donc amenés à considérer des algèbres de termes différentes au cours de ce mémoire.

2.1.2 Définitions et notations générales

Même si l'algèbre de termes considérée pourra varier au cours de ce mémoire, nous utiliserons des notations et des définitions communes.

Les symboles d'arité nulle sont des *constantes*. L'ensemble des variables d'un terme t sera noté $\mathsf{var}(t)$. L'ensemble des noms d'un terme t sera noté $\mathsf{noms}(t)$. Un terme est dit *clos* lorsqu'il

ne contient pas de variables. L'ensemble des termes clos est $T(\mathcal{F},\mathcal{N})$. La *taille* d'un terme t, notée $|t|$ est définie de manière inductive :

$$\begin{cases} |t| = 1 & \text{si } t \text{ est une variable, un nom ou une constante,} \\ |t| = 1 + \sum_{i=1}^{n} |t_i| & \text{si } t = f(t_1, \ldots, t_n) \text{ et } f \text{ symbole d'arité } n. \end{cases}$$

Un terme t est dit *atomique* si t est une constante ou une variable. Si T est un ensemble fini de termes, $|T|$ représente la somme des tailles de ses éléments : $|T| = \sum_{t\in T} |t|$. L'ensemble des sous-termes d'un terme t est noté $\mathsf{st}(t)$ et l'ensemble des sous-termes d'un ensemble de termes T est noté $\mathsf{st}(T)$. Étant donné un terme t et une position p de ce terme, le sous-terme de t à la position p est noté $t|_p$. Lorsqu'on considère la signature $\mathcal{F} = \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, \mathsf{pair}, h, \mathsf{pub}, \mathsf{priv}\}$ (ou un sous-ensemble de celle-ci), on distinguera également les sous-termes *en position de plaintext*. Il s'agit des sous-termes qui peuvent éventuellement accessibles de l'intrus. Formellement, l'ensemble $\mathsf{plaintext}(t)$ des termes en position de plaintext dans un terme t est défini inductivement par :

$$\begin{array}{rcll} \mathsf{plaintext}(\mathsf{pair}(t_1,t_2))) & = & \{\mathsf{pair}(t_1,t_2)\} \cup \mathsf{plaintext}(t_1) \cup \mathsf{plaintext}(t_2) & \\ \mathsf{plaintext}(f(t_1,t_2))) & = & \{f(t_1,t_2)\} \cup \mathsf{plaintext}(t_1) & f \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}\} \\ \mathsf{plaintext}(f(u)) & = & \{f(u)\} & \text{sinon} \end{array}$$

Les substitutions sont notées $\sigma = \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$. Le *domaine* d'une substitution σ est défini par $\mathsf{dom}(\sigma) = \{x_1, \ldots, x_n\}$. Une substitution est dite *close* si tous les t_i sont clos. L'application d'une substitution σ à un terme t est notée $\sigma(t)$ ou $t\sigma$.

Si S est un ensemble de termes et t est un terme, on pourra écrire S, t au lieu de $S \cup \{t\}$.

2.1.3 Déduction

Étant donné un système de déduction tel que celui décrit à la figure 2.1, on dit qu'un terme t est *déductible en un pas* à partir d'un ensemble de termes S s'il existe $t_1, \ldots, t_n \in S$, une substitution θ et une règle de déduction

$$\frac{u_1 \quad \cdots \quad u_n}{u}$$

tels que $t_i = u_i\theta$ pour tout $1 \leq i \leq n$ et $t = u\theta$. Le terme t est *déductible* à partir de S, ce qui est noté $S \vdash t$, s'il existe $t_1, \ldots, t_n \in S$ tels que t_{i+1} est déductible en un pas à partir de $S \cup \{t_1, \ldots, t_i\}$ pour tout $0 \leq i \leq n-1$ et $t_n = t$.

2.1.4 Théories équationelles

Comme évoqué au paragraphe 2.1.1, nous serons amenés, en cours de ce mémoire, à munir l'algèbre de terme $T(\mathcal{F},\mathcal{N},\mathcal{X})$ d'une théorie équationelle E, c'est-à-dire une relation d'équivalence sur les termes, close par application de contexte et par substitution des variables et des noms. Nous écrirons $M =_E N$ si M et N sont équivalents dans E. Une théorie équationelle classique, représentant le déchiffrement et la projection explicites est E_{enc}, induite par les équations ci-dessous :

$$E_{\mathsf{enc}} = \{\mathsf{dec}(\mathsf{enc}(x,y),y) = x, \quad \mathsf{fst}(\mathsf{pair}(x,y)) = x, \quad \mathsf{snd}(\mathsf{pair}(x,y)) = y\}.$$

2.2 Protocoles

De nombreux modèles sont proposés pour les protocoles. Quelques-uns seront décrits au cours de ce mémoire, d'autres ont été mentionnés en introduction. Les notations exactes pour la spécification d'un protocole diffèrent énormément d'un modèle à l'autre. Nous tentons ici d'en décrire les points communs à l'aide d'une notation inspirée du modèle proposé par Mathieu Turuani [RT01, Tur03].

Chaque étape d'un protocole est modélisée par une règle de la forme

$$u \xrightarrow{N_1,\ldots,N_k} v$$

Le terme u représente le message attendu par l'agent actif à cette étape et le terme v représente le message envoyé. Les variables $N_1, \ldots, N_k$ représentent les données générées aléatoirement à cette étape.

Considérons le protocole de Wide Mouthed Frog décrit en introduction (partie 1.2). À la première étape, l'agent A envoie un message pour B, ce qui peut être représenté par la règle

$$\xrightarrow{k_{ab}} \langle a, \langle b, \mathsf{enc}(k_{ab}, k_{as})\rangle\rangle$$

À la deuxième étape, le serveur envoie un message pour B qui dépend du message reçu de A. Il est important de noter que le serveur traitera tout message de la forme $A, B, \{K_{ab}\}_{K_{as}}$, quelle que soit la clef K_{ab}. On utilise donc des variables pour représenter le message attendu pour un agent du protocole.

$$\langle a, \langle b, \mathsf{enc}(x, k_{as})\rangle\rangle \rightarrow \mathsf{enc}(\langle a, x\rangle, k_{bs})$$

Pour un nombre borné de sessions, il suffit de dupliquer les règles autant de fois qu'elles seront jouées et de remplacer les variables $N_1, \ldots, N_k$ par des constantes. Un protocole est donc être simplement décrit par un ensemble fini et partiellement ordonné de règles de la forme $u \rightarrow v$. On retrouve alors exactement le modèle proposé par Mathieu Turuani [RT01, Tur03]. Pour un nombre non borné de sessions, il faut tenir compte de l'état local à chaque session des agents du protocole. Un modèle précis est proposé au chapitre 8 (partie 8.2).

Première partie

Analyse de propriétés d'accessibilité

CHAPITRE 3

Systèmes de contraintes

La propriété la plus étudiée pour les protocoles cryptographiques est certainement celle du secret : peut-on s'assurer qu'un protocole préserve la confidentialité de certaines données sensibles, quelque soit le comportement d'agents malhonnêtes ? Le secret est indécidable lorsqu'on autorise un nombre non borné de sessions [EGS86], même lorsque la taille des messages est bornée [DLMS99, AC02] ou sans considérer de nonces [EGS86].

Lorsque le nombre de sessions est borné, M. Rusinowitch et M. Turuani [RT01, RT03] ont montré que le secret devenait décidable (co-NP-complet). Pour analyser un nombre borné de sessions, J. Millen et V. Shmatikov [MS01] ont introduit les *systèmes de contraintes* qui représentent de manière symbolique l'exécution d'un protocole pour un scénario donné. Dans ce chapitre, nous présentons et revisitons la procédure de décision proposée par Hubert Comon-Lundh [CL04] pour les systèmes de contraintes.

3.1 Modélisation en systèmes de contraintes

La signature considérée est $\mathcal{F} = \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}, h, \mathsf{pair}, \mathsf{pub}, \mathsf{priv}\}$. Les quatre premiers symboles sont d'arité 2, les deux derniers d'arité 1. Comme souvent, nous pourrons écrire $\langle t_1, t_2 \rangle$ au lieu de $\mathsf{pair}(t_1, t_2)$. Les capacités de calcul de l'intrus sont représentées par le système de déduction présenté à la figure 3.1.

3.1.1 Définitions

Nous étudions dans ce chapitre la décidabilité de propriétés d'accessibilité pour un nombre borné de sessions. Borner le nombre de sessions signifie que chaque étape d'un protocole ne sera exécutée qu'un nombre fini de fois. Il est donc possible de *deviner* l'ordre d'exécutions de chacune des étapes. Une fois l'ordre d'exécution fixé, l'existence d'une attaque dépend des messages envoyés par l'intrus à chaque étape. L'ensemble des exécutions possibles peut être représentée de manière élégante à l'aide de systèmes de contraintes.

Définition 3.1 *Un* système de contraintes C *est un ensemble fini d'expressions* $T \Vdash u$, *appelées* contraintes, *où* T *est un ensemble non vide de termes, appelé* membre gauche *de la contrainte et* u *est un terme, appelé* membre droit *de la contrainte, tel que :*

– *les membres gauches des contraintes sont totalement ordonnés par inclusion ;*

$$\frac{S \vdash x \quad S \vdash y}{S \vdash \mathsf{pair}(x,y)} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathsf{enc}(x,y)} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathsf{enca}(x,y)} \qquad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathsf{sign}(x,y)}$$

$$\frac{S \vdash \mathsf{pair}(x,y)}{S \vdash x} \qquad \frac{S \vdash \mathsf{pair}(x,y)}{S \vdash y} \qquad \frac{S \vdash \mathsf{enc}(x,y) \quad S \vdash y}{S \vdash x}$$

$$\frac{S \vdash \mathsf{enca}(x,\mathsf{pub}(y)) \quad S \vdash \mathsf{priv}(y)}{S \vdash x} \qquad \frac{S \vdash \mathsf{sign}(x,\mathsf{priv}(y))}{S \vdash x}\ \textit{(optionnel)} \qquad \frac{}{S \vdash x}\ x \in S$$

FIGURE 3.1 - *Système de déduction de l'intrus.*

– *si* $x \in \mathsf{var}(T)$ *pour une contrainte* $(T \Vdash u) \in C$ *alors*

$$T_x \stackrel{\mathsf{def}}{=} \min\{T' \mid (T' \Vdash u') \in C, x \in \mathsf{var}(u')\}$$

existe et $T_x \subsetneq T$.

Une *solution* d'un système de contraintes C est une substitution θ qui satisfait toutes les contraintes : $T\theta \vdash u\theta$ pour tout $T \Vdash u \in C$.

3.1.2 Exemples

Une fois que le nombre de sessions est fixé et qu'un ordre d'exécution a été choisi pour les règles du protocole, un scénario est une suite de règles de la forme

$$\begin{array}{rcl} u_1 & \rightarrow & v_1 \\ & \vdots & \\ u_n & \rightarrow & v_n \end{array}$$

où les u_i sont des termes représentant la forme des messages attendus par les agents et les v_i sont des termes représentant les messages envoyés. Étant donné un ensemble fini I_0 de termes (clos) représentant la connaissance initiale de l'intrus et k une clef, les attaques menant à la révélation de la clef k sont représentées par les solutions du système de contraintes suivant :

$$\begin{array}{rcl} I_0 & \Vdash & u_1 \\ I_0, v_1 & \Vdash & u_2 \\ & \vdots & \\ I_0, v_1, \ldots, v_{n-1} & \Vdash & u_n \\ I_0, v_1, \ldots, v_{n-1}, v_n & \Vdash & k \end{array}$$

Plus concrètement, considérons le scénario d'attaque décrit en introduction (page 1.2) pour une variante du protocole Wide Mouthed Frog. Considérons $I_0 = \{a, b, c, k_{cs}\}$: l'intrus connaît les noms d'agents a, b, c ainsi que la clef long terme partagée entre c et le serveur.

$$
\begin{array}{lll}
R_1 & C \wedge T \Vdash u \rightsquigarrow C & \text{si } T \cup \{x \mid (T' \Vdash x) \in C, T' \subsetneq T\} \vdash u \\
R_2 & C \wedge T \Vdash u \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash u\sigma & \text{si } \sigma = \mathsf{mgu}(t,u),\ t \in \mathsf{st}(T), \\
 & & \quad t \neq u,\ t,u \text{ non variables} \\
R_3 & C \wedge T \Vdash u \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash u\sigma & \text{si } \sigma = \mathsf{mgu}(t_1,t_2),\ t_1,t_2 \in \mathsf{st}(T), \\
 & & \quad t_1 \neq t_2,\ t_1,t_2 \text{ non variables} \\
R_4 & C \wedge T \Vdash u \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash u\sigma & \text{si } \sigma = \mathsf{mgu}(t_2,t_3),\ \mathsf{enca}(t_1,t_2) \in \mathsf{st}(T), \\
 & & \quad \mathsf{priv}(t_3) \in \mathsf{plaintext}(T) \text{ et } t_2 \neq t_3 \\
R_5 & C \wedge T \Vdash u \rightsquigarrow \bot & \text{si } \mathsf{var}(T,u) = \emptyset \text{ et } T \not\vdash u \\
R_f & C \wedge T \Vdash f(u,v) \rightsquigarrow C \wedge T \Vdash u \wedge T \Vdash v & \text{pour } f \in \{\mathsf{pair}, \mathsf{enc}, \mathsf{enca}, \mathsf{sign}, h\}
\end{array}
$$

FIGURE 3.2 - *Règles de simplification d'un ensemble de contraintes.*

À la première étape, l'agent A envoie un message pour B : la connaissance de l'intrus est augmentée du message $\langle a, \langle b, \mathsf{enc}(k_{ab}, k_{as})\rangle\rangle$. À la deuxième étape, le serveur acceptera toute instantiation θ du terme $\langle a, \langle c, \mathsf{enc}(x, k_{as})\rangle\rangle$ et renverra le message représenté par le terme $\mathsf{enc}(\langle a, x\rangle, k_{cs})\theta$. Le protocole est attaqué si l'intrus peut connaître la clef k_{ab}. L'ensemble des exécutions menant à une attaque est représenté par le système C de contraintes suivant :

$$
\begin{array}{rcl}
I_0, \langle a, \langle b, \mathsf{enc}(k_{ab}, k_{as})\rangle\rangle & \Vdash & \langle a, \langle c, \mathsf{enc}(x, k_{as})\rangle\rangle \\
I_0, \langle a, \langle b, \mathsf{enc}(k_{ab}, k_{as})\rangle\rangle, \mathsf{enc}(\langle a, x\rangle, k_{cs}) & \Vdash & k_{ab}
\end{array}
$$

Une solution de C est la substitution $\theta = \{^{k_{ab}}/_x\}$.

3.1.3 Simplification d'un système de contraintes

La décidabilité du secret pour un nombre borné de sessions se ramène donc à l'existence d'une solution pour un système de contraintes. Il existe des systèmes de contraintes, dits *résolus*, qui admettent toujours une solution.

Définition 3.2 *Un système de contraintes est dit* résolu *si toutes ses contraintes sont de la forme $T \Vdash x$ où x est une variable.*

Tout système résolu C admet une solution : il suffit de choisir $\theta(x) = t \in T_0$ où T_0 est le membre gauche minimal (pour l'inclusion) de C.

J. Millen et V. Shmatikov [MS01] puis Hubert Comon [CL04] ont introduit un ensemble de règles pour transformer pas à pas un système de contraintes en un système de contraintes résolu. Les règles proposées par Hubert Comon [CL04] sont décrites à la figure 3.2. Les règles sont indexées par une substitution, la substitution identité étant implicitement considérée lorsqu'il n'y a pas d'index. On écrit $C \rightsquigarrow^n_\sigma C'$ si $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} C'$ et $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$.

Théorème 3.1 ([CL04]) *L'ensemble de règles décrit à la figure 3.2 est correct, complet et terminant.*

i) (Correction) Si $C \rightsquigarrow^*_\sigma C'$ et θ est solution de C' alors $\sigma\theta$ est solution de C.

ii) (Complétude) Si θ est solution de C alors il existe un système de contraintes C' en forme résolue et des substitutions σ, θ' tels que $\theta = \sigma\theta'$, $C \rightsquigarrow_\sigma^* C'$ et θ' est solution de C'.

iii) (Terminaison) Il n'y a pas de suite infinie $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} C_n \rightsquigarrow_{\sigma_{n+1}} \ldots$

Nous avons montré *[CDD07a, CD09c]* que le système de règles peut être raffiné en évitant d'unifier les paires : les règles restent complètes même si R_2 et R_3 ne sont appliquées que sur des termes non variables et dont le symbole de tête n'est pas pair. Cela nous est utile pour prouver la composabilité des protocoles tagués, résultat qui présenté au chapitre 5 (partie 5.2). Intuitivement, ce raffinement de la procédure de simplification indique qu'il n'est pas nécessaire de taguer les paires pour distinguer deux protocoles (tagués).

3.1.4 Complexité

Le théorème 3.1 permet de conclure à la décidabilité de la propriété du secret. Il ne permet pas immédiatement de retrouver un algorithme co-NP comme celui proposé par M. Rusinowitch et M. Turuani [RT01, RT03]. En effet, l'application des règles de simplification peut créer des branches de longueur exponentielle comme le montre l'exemple suivant :

$$\begin{aligned} T_0 = \{\mathsf{enc}(a, k_0)\} &\Vdash \mathsf{enc}(x_0, k_0) \\ T_1 = T_0 \cup \{\mathsf{enc}(\langle x_0, \langle x_0, a\rangle\rangle, k_1)\} &\Vdash \mathsf{enc}(x_1, k_1) \\ &\vdots \\ T_n = T_{n-1} \cup \{\mathsf{enc}(\langle x_{n-1}, \langle x_{n-1}, a\rangle\rangle, k_n)\} &\Vdash \mathsf{enc}(x_n, k_n) \\ T_{n+1} = T_n \cup \{a\} &\Vdash x_n \end{aligned}$$

Le système de contraintes C admet clairement une solution et C est de taille linéaire en n. On remarque que

$$C \rightsquigarrow_\sigma^{2n} \left\{ \begin{array}{rcl} T_0 & \Vdash & \mathsf{enc}(x_0, K_0) \\ T_{n+1}\sigma & \Vdash & x_n\sigma \end{array} \right.$$

avec $\sigma(x_{i+1}) = \langle x_i, \langle x_i, a\rangle\rangle$ pour $0 \leq i \leq n-1$. Cette dérivation est obtenue en appliquant la règle R_2 puis R_1 pour chaque contrainte $T_i \Vdash \mathsf{enc}(x_i, k_i)$ (avec $1 \leq i \leq n$).

On peut alors montrer par induction sur n qu'il existe une branche de longueur $3(2^n - 1)$ de $T \Vdash x_n\sigma$ à $T \Vdash x_0$ (en forme résolue), où $T = T_{n+1}\sigma$. En utilisant les règles R_{pair} et R_1, on obtient

$$T \Vdash x_n\sigma \overset{R_\langle\rangle}{\rightsquigarrow} \left\{ \begin{array}{rcl} T & \Vdash & x_{n-1}\sigma \\ T & \Vdash & \langle x_{n-1}\sigma, a\rangle \end{array} \right. \rightsquigarrow^m \left\{ \begin{array}{rcl} T & \Vdash & x_0 \\ T & \Vdash & \langle x_{n-1}\sigma, a\rangle \end{array} \right. \overset{R_\langle\rangle}{\rightsquigarrow}$$
$$\left\{ \begin{array}{rcl} T & \Vdash & x_0 \\ T & \Vdash & x_{n-1}\sigma \\ T & \Vdash & a \end{array} \right. \overset{R_1}{\rightsquigarrow} \left\{ \begin{array}{rcl} T & \Vdash & x_0 \\ T & \Vdash & x_{n-1}\sigma \end{array} \right. \rightsquigarrow^m T \Vdash x_0$$

avec $m = 3(2^{n-1} - 1)$ par hypothèse d'induction. La dérivation est donc de longueur exponentielle : $2 \times 3(2^{n-1} - 1) + 3 = 3(2^n - 1)$.

Nous avons montré *[CZ06, CCLZ10]* qu'il n'est en fait jamais utile de considérer à nouveau une contrainte déjà visitée (comme la contrainte $T \Vdash x_{n-1}\sigma$ de notre exemple). À la place de chaque règle de simplification $C \rightsquigarrow_\sigma C'$, nous introduisons la règle suivante, qui évite de revisiter les contraintes déjà explorées :

$$C; D \rightsquigarrow_\sigma C' \setminus D; (C \setminus C') \cup D$$

Les contraintes déjà analysées sont mémorisées dans D. Le système de contrainte initial est alors de la forme $C;\emptyset$. Le nouveau système de règle reste correct et complet et termine cette fois en temps polynomial.

Théorème 3.2 (*[CZ06, CCLZ10]*) *Soit C un système de contraintes et θ une substitution.*

i) (Correction) If $C;\emptyset \rightsquigarrow_\sigma^n C';D'$ et si θ est solution de C' alors $\sigma\theta$ est solution de C.

ii) (Complétude) Si θ est solution de C, alors il existe un système de contraintes C' en forme résolue, un ensemble de contraintes D' et des substitutions σ, θ' tels que $\theta = \sigma\theta'$, $C;\emptyset \rightsquigarrow_\sigma^* C';D'$ et θ' est solution de C'.

iii) (Terminaison) Si $C;\emptyset \rightsquigarrow_\sigma^n C';D'$ alors n est borné par un polynôme en la taille de C.

3.2 Propriétés de sécurité

L'intérêt de la résolution des systèmes de contraintes à l'aide des règles de simplification est multiple. Tout d'abord, c'est un système de règles flexible, auquel il est facile d'ajouter une nouvelle primitive par exemple. Mais surtout l'ensemble des formes résolues forme une représentation symbolique de *toutes* les solutions du système de départ. Aussi, pour décider une propriété donnée sur les traces, il suffit de savoir la décider sur des contraintes résolues. Nous avons utilisé cette approche pour décider plusieurs types de propriétés de sécurité : secret et authentification, cycles de clefs, horodatage ainsi qu'une variante plus forte de la propriété du secret, qui apparaît lorsqu'on introduit les fonctions de hachage.

3.2.1 Propriétés de secret et d'authentification

Pour spécifier des propriétés de type secret et authentification, nous avons proposé *[CCLZ10]* une logique $\mathcal{L}$ définie inductivement de la manière suivante :

$$\phi ::= [m_1 = m_2] \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \bot \mid \top \qquad m_1, m_2 \text{ termes}$$

La sémantique d'une formule close est définie de manière habituelle. En particulier, $\bot$ est évaluée à faux et $\top$ à vrai. Étant donné un système de contrainte C et une formule ϕ de $\mathcal{L}$ telle que $\mathsf{var}(\phi) \subseteq \mathsf{var}(C)$, on dit qu'une θ est *solution* de C pour la propriété ϕ si θ est solution de C et $\phi\theta = \mathbf{true}$.

Pour spécifier le secret, il suffit tout simplement de prendre $\phi = \top$, la formule toujours vraie. Pour l'authentification, la logique $\mathcal{L}$ permet d'exprimer que la valeur reçue par l'agent B (représentée par une variable x) est bien celle envoyée par l'agent A (représentée par une constante k). Il suffit pour cela de considérer la formule $[x = k]$.

La logique $\mathcal{L}$ permet également de considérer des propriétés plus sophistiquées comme le fait que l'agent B a reçu *exactement une fois* la valeur k envoyée par A.

$$\phi_2 = ([x = k] \wedge [y \neq k]) \vee ([x \neq k] \wedge [y = k])$$

Les variables x et y représentent les valeurs reçues par B à la première et à la deuxième session respectivement.

D'autres exemples de propriétés sont décrits dans *[CZ06, CCLZ10]*.

Proposition 3.1 (*[CCLZ10]*) *Soit C un système de contraintes et ϕ une formule de $\mathcal{L}$. Décider si C admet une solution pour ϕ peut être fait en temps polynomial non déterministe.*

3.2.2 Cycles de clefs

La (non) existence de cycles de clefs est une propriété au cœur des preuves de protocoles dans les modèles cryptographiques. En effet, la plupart des schémas de chiffrement sont (prouvés) robustes uniquement dans le cas où les cycles de clefs sont interdits. Il est à noter qu'il n'existe pas d'attaque connue sur les schémas de chiffrement couramment utilisés lorsqu'on crée des cycles de clefs mais aucune preuve de sécurité n'a pu être proposée dans ce cas.

Plusieurs définitions formelles de cycles de clefs ont été proposées. Une des premières définitions proposées par Abadi et Rogaway [AR02] identifie un cycle de clef comme un cycle dans la relation de chiffrement. Ainsi, $\{k\}_k$ et $\{\{a\}_k\}_k$ forment chacun un cycle de clefs. La définition induite par l'approche de Laud [Lau02] correspond à ne considérer de tels cycles que dans les parties *visibles* des messages, c'est-à-dire qui ne sont pas cachées par une clef secrète. Ainsi le message $\{\{k\}_k\}_{k'}$ contient un cycle d'après la définition de Abadi et Rogaway mais n'en contient par pour la définition de Laud, sous réserve que la clef k' soit inconnue de l'intrus. Il suffit souvent de se limiter aux clefs qui apparaissent en position de plaintext. Ainsi $\{k\}_k$ forme un cycle de clef mais pas $\{\{a\}_k\}_k$. Formellement, on dit qu'une clef (atomique) k_1 *chiffre* k_2 dans S, noté $k_1 > k_2$ s'il existe un sous-terme t de S tel que $t = \mathsf{enca}(t', k_1)$ et $k_2 \in \mathsf{plaintext}(t')$. Il existe un *cycle de clef* dans S s'il existe $k_1, \ldots, k_n$ atomiques tels que $k_1 < k_2 < \cdots < k_n < k_1$.

Les preuves de sécurité dans le cas actif demandent souvent une hypothèse plus forte, à savoir que la relation de chiffrement respecte un ordre sur les clefs donné *a priori*. Dans [BP04], l'ordre choisi est par exemple l'ordre dans lequel les clefs sont générées.

Un intrus peut utiliser le protocole pour tenter de créer un cycle de clef. L'existence de cycle de clef est indécidable pour un nombre non borné de sessions *[CZ06, CCLZ10]* mais redevient NP (complet) pour un nombre borné de sessions.

Proposition 3.2 (*[CZ06, CCLZ10]*) *Soit C un système de contraintes et t un terme et $\prec$ un ordre sur les clefs. Décider si C admet une solution θ telle que $t\theta$ admet un cycle de clef ou décider si C admet une solution θ telle que $t\theta$ ne respecte pas l'ordre $\prec$ peut être fait en temps polynomial non déterministe.*

Ce résultat est valable également pour les autres définitions de cycles de clefs évoquées ci-dessus.

3.2.3 Horodatage

Certains protocoles utilisent l'horodatage (ou "timestamps" en anglais) pour s'assurer de la fraîcheur des données reçues. C'est notamment le cas du protocole "Wide-Mouthed-Frog" présenté en introduction. Pour modéliser l'horodatage des données, on considère un sous-ensemble infini Time de noms, qui représente les nombres entiers ou rationnels. Les relations entre les dates sont spécifiées à l'aide de *contraintes de temps entières* (resp. *contraintes de temps rationnelles*) qui sont des conjonctions de formules de la forme

$$\Sigma_{i=1}^{k} \alpha_i x_i \ltimes \beta,$$

où les α_i et β sont des nombres rationnels, $\ltimes \in \{<, \leq\}$, et les x_i sont des variables. Une *solution* de la contrainte de temps entière (resp. rationnelle) T est une substitution close $\sigma = \{^{c_1}/_{x_1}, \ldots, ^{c_k}/_{x_k}\}$, telle que $c_i \in \mathsf{Time}$ et que σ satisfait la contrainte T.

Les contraintes de temps permettent en particulier d'assurer qu'un message reçu portant une date x est suffisamment frais, à l'aide d'une contrainte de la forme $x \geq t$ où $t \in \mathsf{Time}$.

Proposition 3.3 (*[CCLZ10]*) *Soit C un système de contraintes et T une contrainte de temps entière ou rationnelle. Décider si C admet une solution θ telle que $\theta|_{\mathsf{var}(T)}$ soit solution de T peut être fait en temps polynomial non déterministe.*

3.2.4 Secret en présence de hash

La troisième partie de ce mémoire est consacrée à l'étude de la correction des modèles symboliques vis-à-vis des modèles cryptographiques. Il s'agit de transférer les résultats obtenus dans le cadre symbolique : si un protocole est prouvé sûr lorsque les messages sont représentés par une algèbre de termes, qu'en est-il lorsque l'on considère un modèle plus précis, dit cryptographique, où les messages sont représentés par des suites de bits ? La propriété de confidentialité semble particulièrement difficile à transférer. Dans ce chapitre, nous l'avons modélisée comme une propriété d'accessibilité : l'intrus ne doit pas atteindre un état où il connaît le secret. Dans les modèles cryptographiques, une valeur reste confidentielle s'il n'est pas possible d'en obtenir une information partielle. Considérons l'exemple des fonctions de hachage. Un secret s n'est pas déductible (symboliquement) du message $h(s)$. Pourtant, révéler $h(s)$ donne une information partielle sur s. Ainsi, si l'adversaire voit passer le message $h(s_i)$, avec $i \in \{0, 1\}$ et connaît par ailleurs les valeurs s_0 et s_1, il peut calculer $h(s_0)$ et $h(s_1)$, comparer avec la première valeur $h(s_i)$ et déduire quelle valeur a été hachée. La notion de déduction est donc trop faible pour refléter la confidentialité dans les modèles cryptographiques. Nous verrons au chapitre 8 une notion (symbolique) plus forte du secret en présence de fonction de hachage, appelée *secret symbolique* et définie à l'aide de *patterns*. Intuitivement, un protocole préserve le secret symbolique d'une donnée s si celle-ci n'apparaît pas dans la partie *visible* des messages, même lorsque le secret s est connu. En nous ramenant à des systèmes de contraintes résolus, nous avons montré *[CKKW06]* que le secret symbolique reste NP-complet pour un nombre borné de sessions.

3.3 Conclusions et perspectives

Ce chapitre illustre la flexibilité de la résolution des systèmes de contraintes par règles de transformation. Décider une propriété sur les traces d'exécution se ramène à décider cette propriété sur les formes résolues. Nous proposons ci-dessous plusieurs axes de recherche liés à la résolution de systèmes de contraintes (résolus).

3.3.1 Nouvelles propriétés

L'approche présentée dans ce chapitre a été utilisée pour décider des propriétés variées comme le secret, différentes variantes de l'authentification, l'existence de cycles de clefs, les contraintes d'horodatage ou une variante de la propriété de secret dans le cas des fonctions de hachage. Elle a également été utilisée par Detlef Kähler et Ralf Küsters pour analyser les protocoles de signature de contrat [KK05].

Cette approche sera très certainement à nouveau retenue pour d'autres propriétés de sécurité motivées par de nouveaux protocoles. Ainsi, les protocoles de routage sécurisés [PR99, GZA02, HPJ05, BV04] demandent de prendre en compte la topologie du réseau : tous les

nœuds ne sont pas reliés entre eux et un nœud ne peut communiquer qu'avec ses voisins. Ces protocoles de routage doivent notamment assurer qu'un adversaire contrôlant un ou plusieurs nœuds d'un réseau ne peut pas faire accepter une fausse route à un agent (appelé nœud) ou ne peut pas forcer une route à passer par lui. Ces propriétés peuvent s'exprimer comme des prédicats sur les traces et pourront certainement être décidées sur des contraintes résolues, ce qui conduit immédiatement à la décidabilité de ces propriétés pour un nombre borné de sessions.

3.3.2 Fonctions récursives

Les protocoles de routage visent à établir des routes d'un point à un autre du réseau. Chaque nœud ajoute sa connaissance du nœud suivant sous la forme d'une liste chaînée. La source ou la destination de la route (suivant les protocoles) sont amenées à vérifier une liste chaînée de messages. Une route ne sera acceptée que si la liste est bien formée. Ces listes chaînées sont utilisées dans de nombreuses autres applications telles que les chaînes de certificats [GGKL89] ou de manière plus anecdotique, les protocoles de recherche d'offres [KAG98, Rot01]. L'étude de ces protocoles se ramène à l'existence d'une solution pour un système de contraintes

$$C = \left\{ \begin{array}{rcl} I_0 & \Vdash & u_1 \\ I_0, v_1 & \Vdash & u_2 \\ & \vdots & \\ I_0, v_1, \ldots, v_{n-1} & \Vdash & u_n \\ I_0, v_1, \ldots, v_{n-1}, v_n & \Vdash & k \end{array} \right.$$

avec des contraintes d'appartenance $x_i \in \mathcal{L}_i$ pour les variables de C où les $\mathcal{L}_i$ sont des ensembles de termes définis récursivement. Décider l'existence d'une attaque revient à décider l'existence d'une solution pour des contraintes résolues. Il semble possible d'adapter les règles de transformation de manière à éliminer progressivement les contraintes d'appartenance à des langages récursifs. On retrouverait alors des formes résolues classiques.

3.3.3 Propriétés d'équivalence

L'ensemble des formes résolues d'un système de contraintes C permet de représenter symboliquement l'ensemble des solutions de C. Plus formellement, étant donné un système de contraintes C, on peut calculer (en temps exponentiel) l'ensemble $\{C_1, \ldots, C_n\}$ des systèmes de contraintes en forme résolue tels que $C \rightsquigarrow_{\sigma_i} C_i$. Alors

$$\{\theta \mid \theta \text{ solution de } C\} = \cup_{i=1}^{n} \{\sigma_i\theta \mid \theta \text{ solution de } C_i\}.$$

Les propriétés d'équivalences sont des propriétés qui considèrent *l'ensemble* des traces, par opposition aux propriétés d'accessibilité (comme le secret et l'authentification) qui considère les traces une à une. Nous avons vu que la résolution de systèmes de contraintes est très bien adaptée à la décision de propriétés de traces. Cependant, comme l'ensemble des formes résolues caractérise symboliquement l'ensemble des traces, il semble possible de chercher à décider également des propriétés d'équivalences. Nous verrons au chapitre 7 (partie 7.3) que l'équivalence observationelle se ramène à l'égalité des ensembles de traces pour les protocoles déterministes (*[CD09b]*). Pour cette classe de protocoles, l'équivalence observationelle revient donc à tester l'égalité des solutions d'ensembles de contraintes. Pour être précis, le résultat

de *[CD09b]* demande d'enrichir la définition de traces en prenant en compte les *recettes* utilisées par l'intrus pour construire ses messages. Il s'agit donc de tester l'égalité des solutions *au second ordre* de systèmes de contraintes, en reprenant la terminologie utilisée par Mathieu Baudet [Bau05, Bau07].

3.3.4 Théories équationnelles

Une part importante des travaux de recherche sur la vérification symbolique des protocoles cryptographique s'est attachée à considérer des théories équationnelles pour refléter les propriétés algébriques des primitives cryptographiques. Une question naturelle est de savoir comment adapter les règles de transformation lorsque l'algèbre de termes n'est plus une algèbre libre mais une algèbre quotientée par une théorie équationnelle. Ainsi, pour les théories *sous-termes* (Définition 61 du chapitre 6), la résolution des systèmes de contrainte reste un problème NP-complet [DJ04, Bau05, Bau07].

Pour traiter des théories de manière plus générale, une approche consiste à calculer à l'avance les formes normales possibles de manière à s'abstraire de la théorie équationnelle. Cette approche est appelée approche des *variants finis*. Hubert Comon-Lundh et Stéphanie Delaune [CD05, Del06b] ont ainsi proposé une procédure qui permet de réduire l'étude de la satisfaction de contraintes *modulo* une théorie équationnelle à l'étude de la satisfaction de contraintes *modulo* la théorie AC pure (associativité et commutativité). Le prix à payer est la modification du système de déduction de l'intrus. Le problème de la déduction peut alors devenir indécidable. Cette approche a également été utilisée [BCL08] pour montrer la décidabilité de la satisfaction des systèmes de contraintes pour la théorie des signatures en aveugle (définie page 63 du chapitre 6).

Il s'avère cependant difficile d'obtenir des résultats de décidabilité généraux pour des théories AC. Différents fragments décidables ont été proposés pour des théories (avec opérateurs AC) particulières : un fragment décidable de systèmes de contraintes pour la théorie AC pure [BCD07] ; pour la théorie du XOR (ou exclusif) [CKRT03, CLS03] ; plusieurs fragments de la théorie de l'exponentiation modulaire [CKR$^+$03a, MS03] dont un fragment utile pour les protocoles de porte-monnaie électronique *[CD09b]* ; et pour des opérateurs homomorphiques [DLLT06].

Si les théories avec opérateur AC s'avèrent difficiles à traiter, il semble cependant possible de s'attaquer à des théories générales sans opérateur AC comme par exemple celles développées pour traiter du vote électronique (une description précise de ces théories est donnée au chapitre 6, partie 6.5). Au chapitre 6, partie 6.3.2, nous montrons comment revisiter la procédure de décision de Mathieu Baudet [Bau05, Bau07] dans le cadre de l'équivalence statique pour aboutir à un algorithme fonctionnant pour des théories générales (sans opérateur AC). De la même manière, il semble possible de revisiter à nouveau cette procédure de décision dans le cadre des systèmes de contraintes pour traiter des théories plus générales que les théories sous-termes.

CHAPITRE 4

Clauses de Horn

Les clauses de Horn sont bien adaptées pour modéliser l'exécution des protocoles cryptographiques pour un nombre non borné de sessions. C'est en particulier le formalisme au cœur de l'outil ProVerif [Bla01, Bla05, BAF08], largement utilisé pour les protocoles. Dans ce chapitre, nous présentons ce formalisme et discutons ses limites (partie 4.1). Nous proposons un fragment décidable des clauses de Horn, adapté pour les protocoles avec chiffrement CBC ou signature en aveugle (partie 4.2.2). La dernière et principale partie du chapitre est consacrée à l'étude des interfaces de programmation dédiées aux modules matériels de sécurité [BA01, Bon01, Clu03b]. Contrairement aux protocoles, ces interfaces sont sans mémoire (ou presque), ce qui rend la modélisation en clauses de Horn encore plus adaptée.

4.1 Modélisation en clauses de Horn

Dans la première partie de ce chapitre, la signature n'est pas fixée. Notons $\mathcal{F}$ une signature finie quelconque.

4.1.1 Modélisation de l'intrus

Les capacités de déduction de l'intrus s'expriment parfaitement à l'aide de clauses de Horn comme l'illustre l'ensemble de clauses C_{int} présenté à la figure 4.1. Le prédicat unaire I représente la connaissance de l'intrus. L'ensemble (fini) de termes I_0 initialement connu de l'intrus se modélise en considérant l'ensemble de clauses

$$\mathcal{C}_{I_0} = \{I(t) \mid t \in I_0\}$$

On retrouve la notion de déduction habituelle (partie 2.1.3) en considérant qu'un terme t est déductible à partir d'un ensemble (fini) de termes I_0 si et seulement si $I(t)$ appartient au plus petit modèle de Herbrand de $\mathcal{C}_{I_0} \cup C_{\mathsf{int}}$.

4.1.2 Modélisation des protocoles et limites

Une règle de protocole de la forme $u \xrightarrow{N_1,\ldots,N_k} v$ est modélisée en clauses de Horn par la clause

$$I(u) \Rightarrow I(v)$$

$I(x), I(y)$	$\Rightarrow$	$I(\mathsf{pair}(x,y))$	L'intrus peut former la paire de deux messages connus.
$I(x), I(y)$	$\Rightarrow$	$I(\mathsf{enc}(x,y))$	L'intrus peut chiffrer un message connu avec une clef connue.
$I(x)$	$\Rightarrow$	$I(h(x))$	L'intrus peut hacher un message.
$I(\mathsf{pair}(x,y))$	$\Rightarrow$	$I(x)$	L'intrus peut projeter sur la première ou la deuxième composante.
$I(\mathsf{pair}(x,y))$	$\Rightarrow$	$I(y)$	
$I(\mathsf{enc}(x,y)), I(y)$	$\Rightarrow$	$I(x)$	L'intrus peut déchiffrer un message (chiffré avec un algorithme de chiffrement symétrique) s'il possède la clef.
$I(\mathsf{enca}(x,y)), I_{\mathsf{inv}}(y)$	$\Rightarrow$	$I(x)$	L'intrus peut déchiffrer un message (chiffré avec un algorithme de chiffrement asymétrique) s'il possède l'inverse de la clef.
$I(\mathsf{pub}(x))$	$\Rightarrow$	$I_{\mathsf{inv}}(\mathsf{priv}(x))$	Les clefs publiques et privées sont inverses l'une de l'autre.
$I(\mathsf{priv}(x))$	$\Rightarrow$	$I_{\mathsf{inv}}(\mathsf{pub}(x))$	

FIGURE 4.1 - *Ensemble C_{int} de clauses de Horn représentant le pouvoir de l'intrus.*

en remplaçant les variables $N_1, \ldots, N_k$ par des constantes. Ainsi, le protocole de Wide-Mouthed-Frog, présenté en introduction, est modélisé par l'ensemble de clauses ci-dessous :

$$\begin{array}{rcl} & \Rightarrow & \langle a, \langle b, \mathsf{enc}(k_{ab}, k_{as})\rangle\rangle \\ & \Rightarrow & \langle a, \langle c, \mathsf{enc}(k_{ac}, k_{as})\rangle\rangle \\ \langle a, \langle b, \mathsf{enc}(x, k_{as})\rangle\rangle & \Rightarrow & \mathsf{enc}(\langle a, x\rangle, k_{bs}) \\ \langle a, \langle c, \mathsf{enc}(x, k_{as})\rangle\rangle & \Rightarrow & \mathsf{enc}(\langle a, x\rangle, k_{cs}) \end{array}$$

Les deux premières clauses représentent une session où A parle à B (resp. à C). Les deux dernières clauses représentent une session où le serveur transmet un message à B (resp. à C). D'autres clauses peuvent être ajoutées pour considérer le cas où l'agent B jouerait le rôle de l'initiateur par exemple.

On voit ainsi apparaître une première limite de cette modélisation : la fraîcheur des données envoyées (nonces, clefs, etc.) n'est plus assurée : le même nonce peut être envoyé deux fois à des sessions différentes. Il est possible de raffiner la modélisation en représentant les données fraîches non pas par des constantes mais par des termes dépendants du message reçu. La modélisation reste cependant inexacte puisque le même nonce peut être envoyé deux fois à des sessions différentes (dès lors que le message reçu est le même).

Cette abstraction est correcte pour la propriété de secret : si un protocole assure la confi-

dentialité d'une donnée dans le modèle en clauses de Horn, il assure *a fortiori* la confidentialité de la donnée dans un modèle symbolique assurant la fraîcheur des données. Le traitement de l'authentification demande plus d'attention. Bruno Blanchet propose une abstraction correcte de l'authentification à l'aide de prédicats [Bla02].

Une autre limite de la modélisation en clauses de Horn est que l'ordre (partiel) d'exécution des étapes est perdu : il est possible par exemple qu'un agent joue la quatrième étape sans avoir joué la deuxième. Ainsi, les protocoles qui révèlent leurs données fraîches à la fin de chaque session sont très souvent sujet à des fausses attaques avec cette modélisation.

4.2 Décider la satisfiabilité

Malgré les abstractions induites par la modélisation en clauses de Horn, la décision du secret reste indécidable. Il est en effet facile de coder une machine à deux compteurs avec des clauses de la forme

$$I(\{q, x, y\}_k) \Rightarrow I(\{q', x, s(y)\}_k)$$

4.2.1 Résultats existants

Bruno Blanchet a proposé des stratégies de résolution correctes et complètes [Bla01], implémentées dans l'outil ProVerif, mais qui bien sûr ne garantissent pas la terminaison. En pratique l'outil ProVerif s'est révélé très efficace et de nombreux protocoles ont été analysés pour un nombre non borné de sessions à l'aide de cet outil (voir par exemple [ABF04, ABF07, AB05, BC08] pour des protocoles analysés à l'aide de ProVerif).

Des résultats de décidabilité ont également été proposés, en recherchant des fragments décidables de clauses. On peut par exemple remarquer que les clauses de l'intrus (figure 4.1) sont toutes des clauses *plates*. Une clause est dite *plate* si elle est de la forme :

$$\pm I(f(x_1, \ldots, x_n)) \vee \bigvee_{j=1}^{m} \pm I(x_{i_j}).$$

L'ensemble des clauses plates est noté $\mathcal{C}_{\mathcal{I}}$.

On remarque également que les clauses modélisant le protocole Wide-Mouthed-Frog ne comportent qu'au plus une variable. L'ensemble des clauses ne comportant qu'au plus une variable est noté $\mathcal{C}_{\mathcal{P}}$. Elles permettent de modéliser tous les protocoles où, à chaque étape, les agents ne copient ou testent qu'au plus une valeur inconnue (par exemple une clef ou un nonce).

La satisfaction d'un ensemble fini de clauses de $\mathcal{C}_{\mathcal{I}} \cup \mathcal{C}_{\mathcal{P}}$ est 3EXPTIME *[CLC03a]*. La complexité a été ramenée à NEXPTIME par Helmut Seidl et Kumar Verma [SV05, SV08] et DEXPTIME quand les clauses sont de Horn. La satisfiabilité reste également décidable pour les clauses ayant au plus une variable par clause et les clauses "plates" en présence de la théorie équationnelle du "ou exclusif" (définie page 44).

4.2.2 Un nouveau fragment décidable

Nous avons proposé un nouveau fragment décidable adapté à certaines primitives cryptographiques comme le chiffrement CBC ou les signatures en aveugle.

Le chiffrement CBC (Cyphertext Bloc Chaining) permet d'extraire des préfixes des messages chiffrés. Ainsi si la taille de x et y est un multiple de la taille des blocs chiffrés, il est aisé d'extraire $\{x\}_z$ du message $\{x, y\}_z$ sans connaître aucune information sur la clef z. Cette particularité peut être représentée par la clause

$$C_{pre} \stackrel{\text{def}}{=} I(\mathsf{enc}(\mathsf{pair}(x, y), z)) \Rightarrow I(\mathsf{enc}(x, z))$$

De même, les protocoles de vote utilisent des primitives particulières comme les signatures en aveugle. Ainsi, la première étape du protocole de vote FOO 92 [FOO92, KR05] se déroule de la manière suivante : le votant choisit son vote v et le "cache" à l'aide d'un facteur aléatoire r à l'aide d'une primitive proche du chiffrement, notée blind. Il envoie le message $\mathsf{blind}(v, r)$ (avec sa signature) à l'administrateur. L'administrateur vérifie que le votant n'a pas déjà voté et signe le vote : $\mathsf{sign}(\mathsf{blind}(v, r), k_A)$. L'administrateur n'a pas accès au vote exact. Le votant peut alors "retirer" son facteur aléatoire et calculer son vote signé par l'administrateur $\mathsf{sign}(v, k_A)$ grâce aux propriétés particulières de la signature en aveugle. Cette propriété peut être modélisée par la clause.

$$C_{sig} \stackrel{\text{def}}{=} I(\mathsf{sign}(\mathsf{blind}(x, y), z)) \vee I(y) \Rightarrow I(\mathsf{sign}(x, z))$$

Nous avons proposé *[CRZ05]* un nouveau fragment de clauses $\mathcal{C}_S$ telle que la satisfaction d'un ensemble fini de clauses de $\mathcal{C}_\mathcal{I} \cup \mathcal{C}_\mathcal{P} \cup \mathcal{C}_S$ est décidable. $\mathcal{C}_S$ contient en particulier les clauses C_{pre} et C_{sig}. Une définition exacte de $\mathcal{C}_S$ peut être trouvée dans *[CRZ05]*.

4.3 Interfaces de programmation dédiées aux modules de sécurité

La majorité des ordinateurs sont connectés à Internet et peuvent être infectés à l'insu de leur utilisateur, par des virus ou des chevaux de Troie par exemple. Si des données confidentielles sont stockées sur le disque dur d'un ordinateur, il est donc tout à fait possible qu'un virus ait accès à ces données et les transmette à l'extérieur, en utilisant la connexion internet par exemple. Pour protéger les données sensibles, des modules matériels de sécurité ont été développés (Hardware Security Modules - HSM). Ils s'agit de composants matériels, capables d'effectuer des opérations cryptographiques tel que le chiffrement. Leur ouverture provoque l'effacement immédiat des données. Ils ne communiquent avec un ordinateur que de façon limitée, par l'intermédiaire d'une interface de programmation dédiée (Application Programming Interface - API). Les HSMs sont typiquement conçus pour stocker les données et clefs sensibles des utilisateurs. L'utilisateur n'a alors plus d'accès direct à ces données, il connaît seulement des "pointeurs" sur les données (appelés *handles*) et doit faire appel à l'API pour utiliser ses données (pour chiffrer un message à l'aide d'une clef stockée dans un HSM par exemple).

Ces APIs (et les modules de sécurité) sont désormais utilisées dans de nombreuses applications comme les distributeurs de billets ou les systèmes critiques de paiement en ligne. De nouvelles applications comme les passeports biométriques [ecc04] ou la communication entre voitures [RH07] sont en cours de développement.

L'objectif de l'API est (entre autres) de garantir la sécurité des données stockées dans le module HSM, même lorsque l'ordinateur est infecté par un programme malveillant. Les API dédiées aux HSM ont récemment fait l'étude d'analyses plus ou moins formelles. Le standard

$$x, \{y\}_{\mathsf{km}\oplus\mathsf{data}} \rightarrow \{x\}_y \qquad \textbf{(Encipher)}$$

$$\{x\}_y, \{y\}_{\mathsf{km}\oplus\mathsf{data}} \rightarrow x \qquad \textbf{(Decipher)}$$

$$\{y\}_{xkek\oplus xtype}, xtype, \{xkek\}_{\mathsf{km}\oplus\mathsf{imp}} \rightarrow \{y\}_{\mathsf{km}\oplus xtype} \qquad \textbf{(Key Import)}$$

FIGURE 4.2 - *Quelques commandes de l'API IBM 4758 CCA.*

libre PKCS#11 [RSA04] comme des solutions propriétaires telles que l'architecture cryptographique commune d'IBM (IBM 4758 CCA [CCA06]) comportent des failles qui peuvent conduire à la mise en défaut de la politique de sécurité [Bon01, Clu03b, DKS08, LR92]*[CKS07]*. La situation est d'autant plus délicate que la politique de sécurité elle même n'est pas toujours clairement définie [IBM01].

4.3.1 Modélisation des interfaces de programmation

La modélisation en clauses de Horn s'avère particulièrement adaptée au contexte des API dédiées aux HSM. En effet, l'interaction avec le HSM se fait par l'intermédiaire de "commandes" faites à l'API. Des exemples de commandes de l'API IBM 4758 CCA sont décrits à la figure 4.2. La commande (**Encipher**) permet à un utilisateur de chiffrer une donnée x à l'aide d'une clef y stockée dans le HSM. Inversement, la commande (**Decipher**) permet à un utilisateur de déchiffrer un chiffré $\{x\}_y$ à l'aide d'une clef y stockée dans le HSM. L'API IBM 4758 CCA comporte également de nombreuses commandes de gestion de clefs dont la commande (**Key Import**) est un exemple. La commande (**Key Import**) permet à un utilisateur d'importer une clef y reçue d'un autre module, sous un chiffrement à l'aide d'une clef dite de transport $xkek$.

Ces commandes sont *sans mémoire* : les commandes précédentes sont sans influence sur les suivantes, il n'y a pas de sessions. Plus précisément, chaque session comporte exactement une réception et une émission de message. D'autre part, les commandes peuvent être exécutées un nombre arbitraire de fois et dans un ordre quelconque. Les clauses de Horn permettent donc de modéliser parfaitement les commandes des API. Le seul bémol est la possibilité, pour certaines commandes, de créer des nonces. Un traitement particulier sera effectué pour ces commandes.

Les API font un usage intensif du XOR. Nous proposons à la partie 4.3.2 une nouvelle classe décidable de clauses de Horn, avec XOR. Certaines attaques sur les APIs s'appuient sur le fait que le déchiffrement d'un message peut réussir même si la clef de déchiffrement ne correspond pas à la clef de chiffrement. Ces attaques sont appelées "Key conjuring". Nous proposons une modélisation de ce type d'attaque ainsi qu'une nouvelle procédure de décision à la partie 4.3.3. Enfin, plutôt que de vérifier des APIs existants, nous proposons (partie 4.3.4) une API générique permettant d'implémenter la plupart des protocoles existants en garantissant la confidentialité des données stockées sur le HSM.

4.3.2 Une nouvelle classe décidable avec le "ou exclusif"

Les API étudiées dans ce chapitre utilisent principalement du chiffrement symétrique et le "ou exclusif" (XOR). Nous considérerons donc la signature $\mathcal{F} = \{0, \mathsf{enc}, \oplus\}$. Les propriétés

algébriques du XOR, noté $\oplus$, sont modélisées à l'aide de la théorie équationnelle $E_\oplus$ définie par les équations :

$$\begin{array}{rclcrcl} x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & \qquad & x \oplus y & = & y \oplus x \\ x \oplus x & = & 0 & & x \oplus 0 & = & x \end{array}$$

Le pouvoir de l'intrus peut être représenté par les trois clauses suivantes :

$$\begin{array}{rcl} I(x), I(y) & \Rightarrow & I(\mathsf{enc}(x, y)) \\ I(\mathsf{enc}(x, y)), I(y) & \Rightarrow & I(x) \\ I(x), I(y) & \Rightarrow & I(x \oplus y) \end{array}$$

Les commandes des API, en particulier celle de l'API IBM 4758 CCA, sont modélisées à l'aide de clauses de la forme $I(u) \Rightarrow I(v)$ où $u, v \in T(\mathcal{F}, \mathcal{X})$. Malheureusement, ces clauses n'appartiennent pas à la clause décidable proposée dans *[CLC03a]* pour des clauses de Horn avec XOR. Cela est dû au fait que les commandes comportent trop de variables.

Nous avons donc identifié une nouvelle classe de clauses.

Définition 4.1 (Bonne formation) *Un terme t est un terme* XOR *si $t = \bigoplus_{i=1}^{n} u_i$, $n \geq 1$ où chaque u_i est une variable ou une constante.*

Un terme t est un terme chiffré *si $t = \mathsf{enc}(u, v)$ où u et v sont des termes XOR.*

Un terme t est bien formé *si c'est un terme XOR ou un terme chiffré. En particulier, un terme bien formé ne comporte pas de chiffrement imbriqué.*

Une clause $I(t_1), \ldots, I(t_n) \rightarrow I(t_{n+1})$ est bien formée *si*

- *chaque t_i est un terme bien formé ;*
- *$Var(t_{n+1}) \subseteq \bigcup_{i=1}^{n} Var(t_i)$ (aucune variable n'est introduite dans un littéral positif).*

En particulier, les commandes de l'API IBM 4758 CCA peuvent être modélisées à l'aide de clauses bien formées. L'accessibilité (close) des clauses bien formées est décidable.

Théorème 4.1 (*[CKS07]*) *Le problème suivant*

- *Étant donné un ensemble de clauses bien formées $\mathcal{C}$ contenant les clauses $I(x), I(y) \Rightarrow I(x \oplus y)$ et $I(0)$; étant donné un terme clos bien formé u,*
- *est-ce que $\mathcal{C} \cup \{\neg I(u)\}$ est satisfiable ?*

est décidable en temps exponentiel en la taille de $\mathcal{C}$ et de u.

L'argument principal de la preuve est que si $I(u)$ est dérivable à partir de $\mathcal{C}$, alors il existe une dérivation close ne comportant que des termes bien formés.

Implémentation Cette procédure de décision a été implémentée par Gavin Keighren en utilisant une représentation astucieuse des données. Suite à l'attaque découverte par M. Bond [Bon01], trois possibilités (appelées recommandations) ont été proposées par IBM [IBM01] pour contrer l'attaque. À l'aide de l'implémentation de notre procédure, nous avons pu détecter une nouvelle attaque sur la première recommandation d'IBM. Nous avons proposé une petite modification et à nouveau à l'aide de notre implémentation, nous avons pu vérifier que chacune des trois recommandations permettait de rétablir la sûreté de l'API (en ce qui concerne la confidentialité des codes secrets des utilisateurs - codes PIN).

4.3.3 Key conjuring

4.3.3.1 Qu'est-ce que le Key conjuring ?

Certains types d'attaques demandent une modélisation plus fine des APIs. Ainsi, il est possible de remplacer avec succès des chiffrés valides par des nombres aléatoires. En effet, le chiffrement utilisé dans une API comme celle d'IBM 4758 CCA est le DES (et triple DES). Une clef (simple) est de taille 64 bits au total qui sont divisés en 8 groupes. Chaque groupe comporte 7 bits de clef et un dernier bit appelé *bit de parité*. Pour les *clefs paires*, chaque bit de parité doit être choisi de manière à ce que la parité de son groupe soit paire. De manière similaire, pour les *clefs impaires*, chaque bit de parité doit être choisi de manière à ce que la parité de son groupe soit impaire. Lorsque le HSM déchiffre une clef, il en vérifie la parité.

Dans la section précédente, nous avons représenté la commande de chiffrement (**Encipher**) par la clause

$$I(x), I(\{y\}_{\mathsf{km}\oplus\mathsf{data}}) \Rightarrow I(\{x\}_y).$$

Le déchiffrement est représenté de manière implicite. Cela suggère que la commande ne peut réussir que si l'adversaire connaît un chiffré de la forme $\{k\}_{\mathsf{km}\oplus\mathsf{data}}$. En réalité, un attaquant peut tout simplement deviner un nombre n de 64 bits et l'utiliser à la place du chiffré. Le HSM déchiffrera alors n avec la clef $\mathsf{km} \oplus \mathsf{data}$ et vérifiera la parité du résultat. En général, la parité sera invalide (tous les blocs ne seront pas de la même parité) et la clef sera rejetée. Mais si l'attaquant choisit n de manière aléatoire, le test réussira avec probabilité $\frac{1}{256}$. Il peut sembler peu utile de faire accepter par le HSM une clef inconnue de l'intrus mais cette technique, appelée "Key conjuring" [Bon01], a été utilisée pour mettre en œuvre plusieurs attaques [Bon01, Clu03b, CB03].

4.3.3.2 Modélisation

Nous considérons une signature avec sorte avec un opérateur explicite dec pour le déchiffrement :

$$\begin{array}{rcccccc} \oplus & : & \mathsf{Base} & \times & \mathsf{Base} & \rightarrow & \mathsf{Base} \\ \mathsf{enc} & : & \mathsf{Base} & \times & \mathsf{Base} & \rightarrow & \mathsf{Cipher} \\ \mathsf{dec} & : & \mathsf{Cipher} & \times & \mathsf{Base} & \rightarrow & \mathsf{Base} \end{array}$$

La théorie $E_\oplus$ est augmentée des équations

$$\mathsf{dec}(\mathsf{enc}(x,y),y) = x \qquad \mathsf{enc}(\mathsf{dec}(x,y),y) = x$$

pour former la théorie E_{API}.

Les tests de parité sont représentés par les prédicats chkEven et chkOdd, accompagnés des clauses

$$\begin{array}{rcl} \mathsf{chkEven}(x_1),\ \mathsf{chkEven}(x_2) & \Rightarrow & \mathsf{chkEven}(x_1 \oplus x_2) \\ \mathsf{chkOdd}(x_1),\ \mathsf{chkOdd}(x_2) & \Rightarrow & \mathsf{chkEven}(x_1 \oplus x_2) \\ \mathsf{chkEven}(x_1),\ \mathsf{chkOdd}(x_2) & \Rightarrow & \mathsf{chkOdd}(x_1 \oplus x_2) \end{array}$$

Définition 4.2 (clause d'API) *Une clause d'API est une clause de la forme*

$$\mathsf{chk}_1(u_1), \ldots, \mathsf{chk}_k(u_k), I(x_1), \ldots, I(x_n) \Rightarrow I(t)$$

telle que

- $x_1, \ldots, x_n$ *sont des variables ;*

- t *est un terme tel que* $\mathsf{var}(t) \subseteq \{x_1, \ldots, x_n\}$ *et tel que* t *est un terme* pur *: il ne contient qu'au plus un symbole de chiffrement qui dans ce cas doit apparaître en tête ;*
- $u_1, \ldots, u_k$ *sont des termes de sorte* Base *dont le symbole de tête n'est pas* $\oplus$*,*
- $\mathsf{chk}_i \in \{\mathsf{chkOdd}, \mathsf{chkEven}\}$*,* $1 \leq i \leq k$*.*

Les commandes de l'API IBM 4758 CCA ainsi que les clauses de l'intrus peuvent être représentées par des clauses d'API. Ainsi, la commande (**Encipher**) est représentée par la clause

$$C_{(\mathbf{Encipher})} \stackrel{\mathsf{def}}{=} \mathsf{chkOdd}(\mathsf{dec}(y, \mathsf{km} \oplus \mathsf{data})),\ I(x),\ I(y)\ \Rightarrow\ I(\mathsf{enc}(x, \mathsf{dec}(y, \mathsf{km} \oplus \mathsf{data})))$$

Nous avons proposé *[CDS07a]* une transformation générique pour déduire, à partir d'un ensemble $\mathcal{C}$ de clauses modélisant une API, un ensemble de règles $\mathsf{KeyCj}(\mathcal{C})$ représentant les tentatives possibles de "Key Conjuring". Ainsi, la transformation de la clause représentant la commande (**Encipher**) résulte en la règle $\mathsf{KeyCj}(C_{(\mathbf{Encipher})})$ définie par

$$I(x) \stackrel{\mathsf{new}\, n}{\Rightarrow} I(\mathsf{enc}(x, \mathsf{dec}(n, \mathsf{km} \oplus \mathsf{data}))),\ I(n),\ \mathsf{chkOdd}(\mathsf{dec}(n, \mathsf{km} \oplus \mathsf{data}))$$

Ces nouvelles règles ne sont plus des clauses puisqu'elles comportent des nonces frais, représentés par $\mathsf{new}\ n$. De plus, il ne serait pas raisonnable d'autoriser l'attaquant à réaliser un nombre arbitraire de Key Conjuring puisque chaque Key Conjuring ne réussit qu'avec probabilité $\frac{1}{256}$. Aussi, nous ne considérerons qu'un nombre borné d'application des règles de key conjuring.

4.3.3.3 Décidabilité

La sécurité d'une API, en considérant les attaques par Key Conjuring se définit (informellement) désormais de la manière suivante :

> Étant donné un ensemble $\mathcal{C}$ de clauses d'API, étant donné un terme t représentant une donnée confidentielle et un entier k, est-ce que $I(t)$ est déductible à partir de $\mathcal{C}$ en utilisant au plus k applications des règles de $\mathsf{KeyCj}(\mathcal{C})$ (ce qui est noté $\mathcal{C} \vdash_{E_{\mathsf{API}}}^{\mathsf{KeyCj}(\mathcal{C}) \leqslant k}$) ?

Une définition formelle est bien sûr donnée dans *[CDS07a]*. L'entier k est un paramètre fixé par l'utilisateur suivant le niveau de sécurité souhaité.

Théorème 4.2 (*[CDS07a]*) *Soit* $\mathcal{C}$ *un ensemble de clauses d'API bien formées,* t *un terme pur et* k *un entier. Le problème* $\mathcal{C} \vdash_{E_{\mathsf{API}}}^{\mathsf{KeyCj}(\mathcal{C}) \leqslant k}$ *est décidable (en temps non déterministe doublement exponentiel).*

La notion de bonne formation diffère de la définition 4.1. Elle assure que la parité des termes utilisés en position clef est toujours testée et que les clefs de déchiffrement sont construites de manière unique (pour tous $\mathsf{dec}(x, v_1)$, $\mathsf{dec}(x, v_2)$ sous-termes de $\mathcal{C}$, les termes v_1 et v_2 doivent être égaux syntaxiquement).

La preuve de ce résultat fait en particulier appel à la technique des variants finis [CD05, Del06b] pour deviner à l'avance les formes normales des termes utilisés au cours d'une attaque.

4.3.4 Une interface de programmation sûre et générique

Nous avons mentionné dans les parties précédentes que les APIs existantes (telles que PKCS#11 [RSA04] l'API IBM 4758 CCA [CCA06]) comportent souvent des failles. De plus,

elles sont développées pour des utilisations fixées à l'avance et devront être reprogrammées si de nouvelles applications se révèlent utiles.

Nous avons proposé *[CS09a]* une API générique qui permet d'implémenter de manière sûre la plupart des protocoles. L'idée principale est que lorsque qu'une donnée sensible (une clef ou un nonce par exemple) est transmise sur le réseau, la politique de sécurité attachée à la donnée doit être transmise dans le même message.

Algèbre de termes Nous considérons à nouveau une algèbre de terme avec sortes

$$\begin{aligned} \mathsf{Keyv} &::= \mathsf{Key} \mid \mathsf{VarKey} \\ \mathsf{Noncev} &::= \mathsf{Nonce} \mid \mathsf{VarNonce} \\ \mathsf{Msg} &::= \mathsf{Agent} \mid \mathsf{Keyv} \mid \mathsf{Noncev} \mid \mathsf{Var} \mid \{\mathsf{Msg}\}_{\mathsf{Keyv}} \mid \mathsf{pair}(\mathsf{Msg},\mathsf{Msg}) \\ \mathsf{Handle} &::= h_a^{\alpha}(\mathsf{Nonce},\mathsf{Msg},i,S) \end{aligned}$$

où $i \in \{0,1,2,3\}, S \subseteq \mathsf{Agent}, a \in \mathsf{Agent}, \alpha \in \{r,g\}$.

La particularité de cette algèbre est qu'elle permet de modéliser explicitement les pointeurs (handles) fournis à l'utilisateur par l'API pour qu'il puisse manipuler les données stockées dans le HSM sans les connaître explicitement. Le handle $h_a^{\alpha}(n,k,i,S)$ représente un pointeur pour une donnée k stockée sur le HSM de l'agent a. L'ensemble S représente les agents autorisés à accéder à k. Le nonce n permet de distinguer différents handles qui peuvent faire référence à la même donnée. L'étiquette α permet à l'API de distinguer les données k générées par elle-même ($\alpha = g$) des données k reçues de l'extérieur ($\alpha = r$). Nous verrons à la fin de cette partie que cette distinction permet à l'API de se prémunir des attaques par rejeu. Enfin, l'entier i indique le niveau de sécurité de la donnée k. Nous considérons quatre niveaux de sécurité :

- 0 : donnée publique
- 1 : donnée secrète qui n'est pas utilisée pour chiffrer (en particulier, un nonce)
- 2 : clefs court-terme
- 3 : clefs long-terme

Nous considérons l'ensemble $\mathcal{P} = \{k_a \mid a \in \mathsf{Agent}\} \cup \{I\}$ de prédicats. Le prédicat K_a avec $a \in \mathsf{Agent}$ représente la connaissance d'un agent a. Le prédicat I représente à nouveau la connaissance de l'intrus.

Description de l'API Notre API comporte simplement trois règles génériques.

Génération de données. Un agent a peut demander à l'API de générer une donnée K fraîche de niveau de sécurité $i \in \{0,1,2\}$, à l'attention du groupe d'agent S.

$$\overset{N,K}{\Rightarrow} K_a(h_a^g(N,K,i,S)) \qquad i \geqslant 1 \tag{4.1}$$

$$\overset{N,K}{\Rightarrow} K_a(K), K_a(h_a^g(N,K,0,\emptyset)) \tag{4.2}$$

L'agent reçoit en retour un handle sur la donnée K ainsi que la donnée elle-même si le niveau de sécurité demandé est 0 (donnée publique).

Chiffrement de données. Un agent a peut demander à l'API de chiffrer des données publiques $x_1,\ldots,x_k$ ainsi que des données secrètes $y_1,\ldots,y_l$ à l'aide d'une clef K. L'agent a ne connaît K que par l'intermédiaire d'un handle $h_a^{\alpha}(X_n,K,i_0,S_0)$ et ne connaît les données y_j

que par l'intermédiaire d'un handle $h_a^\alpha(X_{n_j}, y_j, i_j, S_j)$.

$$\begin{array}{l} K_a(h_a^\alpha(X_n, K, i_0, S_0)), \\ K_a(x_1), \ldots, K_a(x_k), \\ K_a(h_a^\alpha(X_{n_1}, y_1, i_1, S_1)), \ldots, K_a(h_a^\alpha(X_{n_l}, y_l, i_l, S_l)) \\ \Rightarrow K_a(\{0, x_1, \ldots, 0, x_k, y_1, i_1, S_1, \ldots, y_l, i_l, S_l\}_K) \end{array} \tag{4.3}$$

Intuitivement, l'API chiffre comme demandé les données en ajoutant dans le message chiffré les informations concernant le niveau de sécurité ainsi que les groupes d'agents habilités à lire chacune des clef. La règle est exécutée seulement si $i_0 > i_j$ pour assurer que la clef de chiffrement est de niveau suffisant et si $a \in S_0 \subseteq S_j$ pour assurer que les données ne pourront être lues que par des personnes habilitées.

Déchiffrement de données. Un agent a peut demander à l'API de déchiffrer un message reçu à l'aide d'une clef K.

$$\begin{array}{c} K_a(h_a^\alpha(X_n, K, i_0, S_0)), K_a(\{0, x_1, \ldots, 0, x_k, y_1, i_1, S_1, \ldots, y_l, i_l, S_l\}_K) \overset{N_1,\ldots,N_l}{\Rightarrow} \\ K_a(x_1), \ldots, K_a(x_k), \\ K_a(h_a^r(X_{n_1}, y_1, i_1, S_1)), \ldots, K_a(h_a^r(X_{n_l}, y_l, i_l, S_l)) \end{array} \tag{4.4}$$

Il reçoit alors en clair les données publiques et un handle pour les données secrètes contenues dans le message. La règle est exécutée seulement si $i_0 > i_j$ et si $a \in S_0 \subseteq S_j$.

Pour les commandes de chiffrement comme celles de déchiffrement, les éléments (publics ou privés) peuvent être bien sûr placés dans n'importe quel ordre, au choix de l'utilisateur. Pour simplifier la présentation, nous n'avons modélisé ici que le cas où les données publiques sont placées en tête. L'ensemble de ces règles est noté API.

Un état du système est donné par une famille $\{S_b \mid b \in \mathsf{Agent} \cup \{\mathsf{int}\}\}$. Chaque S_b, $b \in \mathsf{Agent}$, représente la connaissance locale d'un agent b. L'ensemble de termes S_{int} représente la connaissance courante de l'intrus. L'évolution d'un état du système $S \to S'$ est obtenu en appliquant une règle de la forme $P_1(u_1), \ldots, P_k(u_k) \overset{N_1,\ldots,N_p}{\longrightarrow} Q_1(v_1), \ldots, Q_l(v_l)$ et en respectant la fraîcheur des données $N_1, \ldots, N_p$. La sémantique précise de $S \to_{\mathcal{R}} S'$ pour un ensemble de règles $\mathcal{R}$ est décrite dans *[CS09a]*.

Implémentation Notre API permet d'implémenter la plupart des protocoles à clefs symétriques, en particulier tout ceux mentionnés dans la bibliothèque de Clark-Jacob (partie 6.3) [CJ97]. L'intérêt de l'API est alors de permettre à l'utilisateur d'utiliser des protocoles existants tout en assurant la sécurité de ses clefs même lorsque sa machine est infectée par un virus qui peut lui aussi interagir avec l'API.

Nous avons proposé *[CS09a]* un algorithme pour déduire automatiquement d'un protocole la liste de commandes à effectuer sur l'API. Cet algorithme a été implémenté par Graham Steel en Prolog.[1]

Ainsi, le protocole de Wide-Mouthed-Frog présenté en introduction peut être implémenté de la manière suivante. Initialement, les agents connaissant les clef long termes K_{as} par l'inter-

[1] Le code source ainsi que les résultats sur la bibliothèque de Clark-Jacob sont accessibles sur la page `http://www.lsv.ens-cachan.fr/~steel/GenericAPI/`

médiaire d'un pointeur de la forme $h_a^r(N_1, K_{as}, 3, \{A, S\})$. Pour la première règle de l'agent A :

$$A \to S : A, B, \{K_{ab}\}_{K_{as}}$$

il suffit à l'agent de demander à son API de lui générer une clef de session :

$$\overset{N,K_{ab}}{\Rightarrow} K_a(h_a^g(N, K_{ab}, 2, \{A, B, S\}))$$

L'agent obtient alors un pointeur $h_a^g(N, K_{ab}, 2, \{A, B, S\})$ sur la clef K_{ab}. Il demande ensuite à l'API de construire le message $\{K_{ab}\}_{K_{as}}$ pour lui à l'aide de la commande de chiffrement :

$$K_a(h_a^r(N_1, K_{as}, 3, \{A, S\})),\ K_a(h_a^g(N, K_{ab}, 2, \{A, B, S\})) \Rightarrow K_a(\{K_{ab}, 2, \{A, B, S\}\}_{K_{as}})$$

Cette règle est possible car la clef K_{as} est de niveau strictement supérieure à K_{ab} et parce que $\{A, S\} \subseteq \{A, B, S\}$. Il suffit alors à l'agent A de concaténer les identités A et B et d'envoyer le message au serveur.

Pour la première règle du serveur :

$$S \to B : \{A, K_{ab}\}_{K_{bs}}$$

le serveur commence par déchiffrer le message reçu à l'aide de la commande de déchiffrement :

$$K_s(h_s^r(N_2, K_{as}, 3, \{A, S\})),\ K_s(\{K_{ab}, 2, \{A, B, S\}\}_{K_{as}}) \overset{N_3}{\Rightarrow} K_s(h_s^r(N_3, K_{ab}, 2, \{A, B, S\}))$$

Il demande ensuite à l'API de construire le message $\{A, K_{ab}\}_{K_{bs}}$ pour lui à l'aide de la commande de chiffrement :

$$K_s(h_s^r(N_2, K_{as}, 3, \{A, S\})),\ K_s(A),\ K_s(h_s^r(N_3, K_{ab}, 2, \{A, B, S\})) \Rightarrow K_s(\{0, A, K_{ab}, 2, \{A, B, S\}\}_{K_{as}})$$

Cette implémentation modifie légèrement le protocole puisque des annotations sont ajoutées pour chaque donnée. Ajouter explicitement le nom des agents à qui les données sont destinées fait partie des recommandations usuelles pour la construction de protocoles [AN96] et renforce leur sécurité.

Sécurité On distingue un ensemble $H \subseteq \mathsf{Agent}$ d'agents dits honnêtes. La sécurité se déduit des informations fournies dans les handles : l'intrus ne doit pas avoir accès au contenu d'un handle s'il n'y est pas autorisé.

$$\forall S \subseteq H \qquad \neg I(y) \vee \neg I(h_a^\alpha(x, y, i, S)) \qquad (\mathbf{Sec})$$

À l'inverse, l'intrus a accès aux HSM des agents corrompus et peut interagir avec les APIs de tous les agents (honnêtes ou corrompus).

$$K_a(x) \Rightarrow I(x) \tag{4.5}$$
$$I(x) \Rightarrow K_a(x) \tag{4.6}$$
$$K_b(h_b^\alpha(x, y, i, S)) \Rightarrow I(y) \qquad b \notin H \tag{4.7}$$

L'ensemble de ces trois règles est noté $\mathsf{CONTROL}$.

Théorème 4.3 (*[CS09a]*) *La propriété* **Sec** *est préservée par application des règles de* $\mathsf{API} \cup C_{\mathsf{int}} \cup \mathsf{CONTROL}$.[2]

Ce théorème assure la confidentialité des données stockées par l'API, dès lors que le système satisfait initialement la propriété **Sec**. Cela est vrai en particulier à l'initialisation du système : la connaissance initiale des agents ne contient que des handles et que la connaissance initiale de l'intrus ne contienne que des données atomiques, qui n'apparaissent pas dans la connaissance des agents.

Nous avons étendu ce résultat au cas où l'intrus est capable d'apprendre des clefs anciennes *[CS09a]*. L'intrus connaît alors à la fois un handle valide $h_a^\alpha(n, k, i, S)$ et une clef k pour S ensemble d'agents honnêtes et $i \geq 1$. Pour préserver la sécurité des données, il faut alors restreindre l'application du déchiffrement aux cas où le chiffré contient au moins une donnée (fraîche) engendrée par l'API (ce qui est modélisé par l'étiquette r du handle correspondant). De manière remarquable, nous avons pu constater que cette restriction ne permet plus d'implémenter la version du protocole de Wide-Mouthed-Frog présenté en introduction, ni la version initiale du protocole de Needham-Schroeder à clefs symétriques ni le protocole de Yahalom. Il se trouve en fait que ces trois protocoles sont sujets à des attaques par rejeu [CJ97, PS00] et sont donc mis en défaut dès que l'attaquant peut apprendre des clefs anciennes.

4.4 Conclusion et perspectives

Il s'avère assez difficile de mettre au point de nouveaux fragments décidables pour les clauses de Horn même si des applications précises comme les signatures en aveugles ou les interfaces de programmation dédiées aux modules de sécurité ont motivé de nouveaux résultats présentés au cours de ce chapitre.

En revanche, le développement des APIs pour les modules de sécurité amène des problématiques nouvelles et prometteuses qui débordent du cadre des clauses de Horn. Nous avons ainsi proposé des techniques d'analyse ainsi qu'une API générique mais de nombreux développements sont à envisager, que nous décrivons ci-dessous.

Les modèles considérés au cours de ce chapitre se limitent au chiffrement symétrique. Il parait naturel et important de considérer d'autres primitives comme les fonctions de hachages ou les signatures. Le chiffrement asymétrique peut également être envisagé. Il permettrait d'implémenter de plus nombreux protocoles mais il peut être trop coûteux en ressources (temps et capacité de calcul) pour à mis en en œuvre dans toutes les situations.

Certaines attaques (comme le "Key Conjuring") sont liées aux détails de l'implémentation des primitives de chiffrement. Il serait intéressant d'étudier si des garanties cryptographiques (comme celles développées aux chapitres 8 et 9) peuvent être obtenues sous des hypothèses réalistes au regard des contraintes liées aux APIs.

Certaines applications parmi les plus récentes visent à développer des APIs dédiées aux voitures [RH07]. Elles apportent de nouvelles problématiques qui n'ont pas été considérées jusqu'ici comme le respect de la vie privée des conducteurs ou la prise en compte du déplacement du véhicule au cours des communications.

[2] L'ensemble C_{int} a été défini page 40.

CHAPITRE 5

Synthèse et combinaison sûres de protocoles

Plutôt que de développer des techniques de vérification pour analyser des protocoles existants, il est possible de proposer des techniques pour produire des protocoles sûrs *par construction*. Dans un article relativement ancien [AN96], Martín Abadi et Roger Needham ont proposé des principes généraux pour concevoir des protocoles fiables. Dans le contexte de la cryptographie, plusieurs résultats visent à proposer des *compilateurs* de protocoles : Goldreich, Micali et Wigderson [GMW87] ont montré comment compiler des protocoles sûrs lorsque les agents suivent honnêtement le protocole (mais peuvent tenter d'apprendre des informations auxquelles ils n'ont pas accès) en des protocoles sûrs quel que soit le comportement des agents. Bellare, Canetti et Krawczyk [BCK98] ont montré comment transformer un protocole sûr lorsque les communications entre les parties sont authentifiées en un protocole sûr sans cette hypothèse. Pedro Adao et Cédric Fournet [FA06] ont proposé un calcul proche du pi-calcul, sans primitives cryptographiques, qui peut-être compilé au niveau cryptographique en préservant la sûreté du protocole. Plusieurs techniques ont également été proposées pour permettre le développement modulaire de protocoles. Ainsi, Mitchell *et al.* [DMP01, DMP03, DDMP05] ont proposé une méthodologie pour développer les protocoles de façon modulaire lorsque les propriétés de sécurité sont ajoutées peu à peu. Joshua Guttman [GT00, Gut04, Gut09] a exploré des conditions suffisantes pour permettre la composition sûre de protocoles.

Dans ce chapitre, nous présentons des techniques possibles pour construire des protocoles sûrs. Dans la première partie, nous proposons une transformation simple et générique pour construire un protocole sûr, contre un attaquant actif et un nombre arbitraire de sessions, à partir d'un protocole sûr pour une seule session, sans attaquant. L'avantage de cette approche est de proposer une construction générique qui permet à l'utilisateur de s'abstraire des problèmes de sécurité. Dans la deuxième partie de ce chapitre, nous étudions la question de la composition des protocoles : étant donné un protocole dont on a prouvé la robustesse, à quelle condition peut-on l'utiliser en même temps que d'autres protocoles, partageant certaines de ses clefs ? Nous proposons une condition suffisante simple pour permettre le partage de clefs entre différents protocoles tout en préservant la sécurité.

Ces deux résultats s'énoncent facilement dans un cadre relativement informel ; aussi nous n'entrerons pas dans les détails dans ce chapitre. Les preuves des résultats avancés ont bien sûr été effectuées pour des modèles définis précisément.

5.1 Synthèse sûre de protocoles

Nous avons proposé *[CWZ07b]* une transformation simple et intuitive qui permet de garantir la sécurité d'un protocole contre un attaquant actif et pour un nombre arbitraire de sessions. Notre transformation peut être vue comme un compilateur : le protocole d'entrée spécifie le comportement souhaité, sans même avoir besoin de primitives cryptographiques. Il suffit qu'il soit sûr pour une seule session et sans attaquant (pas même un intrus passif qui pourrait lire les messages envoyés). Le protocole en sortie est obtenu de manière automatique à partir du premier, en ajoutant des primitives cryptographiques, de manière à obtenir un protocole sûr. La transformation comporte deux étapes. Dans un premier temps, les agents génèrent dynamiquement un identifiant unique de session. Puis, dans un deuxième temps, les messages du protocole sont munis de l'identifiant de session, sont signés par l'émetteur et chiffrés avec la clef publique du destinataire.

Notre transformation est inspirée d'un compilateur introduit par Katz et Yung [KY03], qui transforme un protocole sûr d'échange de clefs contre un intrus passif en un protocole sûr contre un intrus actif. Leur transformation est plus simple que la nôtre car les messages du protocole transformé n'ont pas besoin d'être (re)chiffrés. Par contre, elle demande une hypothèse de sécurité plus forte sur le protocole de départ et ne s'applique qu'aux protocoles d'échange de clefs.

5.1.1 Transformation générique

Nous détaillons ci-dessous le fonctionnement de notre transformation. Considérons un protocole à k participants $A_1, \ldots, A_k$ et n échanges de messages.

$$\begin{aligned} A_{i_1} &\rightarrow A_{j_1} : \quad m_1 \\ &\vdots \\ A_{i_n} &\rightarrow A_{j_n} : \quad m_n \end{aligned}$$

Le protocole transformé commence par une phase préliminaire où chaque participant A_i envoie un nonce frais N_i à tous les autres participants.

$$\begin{aligned} A_1 &\rightarrow \mathsf{All} : \quad N_1 \\ &\vdots \\ A_k &\rightarrow \mathsf{All} : \quad N_k \end{aligned}$$

L'implémentation exacte de cette première phase n'a pas d'importance. On pourrait également supposer par exemple que les agents se transmettent les nonces de proche en proche :

$$\begin{aligned} A_1 &\rightarrow A_2 : \quad N_1 \\ A_2 &\rightarrow A_3 : \quad N_1, N_2 \\ &\vdots \\ A_{k-1} &\rightarrow A_k : \quad N_1, N_2, \ldots, N_{k-1} \\ A_k &\rightarrow \mathsf{All} : \quad N_1, N_2, \ldots, N_{k-1}, N_k \end{aligned}$$

La concaténation des nonces avec les identités des participants forme l'*identifiant de sessions* $\mathsf{sessionID} = \langle A_1, A_2, \ldots, A_k, N_1, N_2, \ldots, N_k \rangle$.

L'adversaire peut parfaitement interférer lors de cette première phase et, par exemple, intercepter et modifier certains nonces. Un tel comportement sera détecté lors de la seconde phase. La suite du protocole fonctionne à peu près comme le protocole original à ceci près que chaque message est envoyé avec l'identifiant de sessions, le tout signé par l'émetteur et chiffré par la clef publique du destinataire

$$\begin{aligned} A_{i_1} \to A_{j_1} : &\quad \{m_1, \mathsf{sign}(\langle m_1, p_1, \mathsf{sessionID}\rangle, \mathsf{sk}(A_{i_1}))\}_{\mathsf{pub}(A_{j_1})} \\ &\vdots \\ A_{i_n} \to A_{j_n} : &\quad \{m_n, \mathsf{sign}(\langle m_n, p_n, \mathsf{sessionID}\rangle, \mathsf{sk}(A_{i_n}))\}_{\mathsf{pub}(A_{j_n})} \end{aligned}$$

Les constantes p_j représentent les points de contrôle (le numéro d'étape) du programme correspondant à l'émetteur A_{i_j} du message. Le message $\mathsf{sign}(m, \mathsf{sk}(A))$ représente le message m signé par A et le message $\{m\}_{\mathsf{pub}(A)}$ représente le message m chiffré par la clef publique de A.

5.1.2 Exemple

Reprenons l'exemple du protocole Wide Mouthed Frog, présenté en introduction. Après transformation, le protocole est :

$$\begin{aligned} A \to \mathsf{All} : &\quad N_A \\ S \to \mathsf{All} : &\quad N_S \\ B \to \mathsf{All} : &\quad N_B \\ A \to S : &\quad \{A, \{B, K_{ab}\}_{K_{as}}, \mathsf{sign}(\langle A, \{B, K_{ab}\}_{K_{as}}, 1, \mathsf{sessionID}\rangle, \mathsf{sk}(A))\}_{\mathsf{pub}(S)} \\ A \to B : &\quad \{\{A, K_{ab}\}_{K_{bs}}, \mathsf{sign}(\langle\{A, K_{ab}\}_{K_{bs}}, 1, \mathsf{sessionID}\rangle, \mathsf{sk}(S))\}_{\mathsf{pub}(B)} \end{aligned}$$

où $\mathsf{sessionID} = \langle A, S, B, N_A, N_S, N_B\rangle$.

5.1.3 Sécurité

Notre transformation permet de transférer de nombreuses propriétés satisfaites par le protocole initial (pour une session unique et sans attaquant) même dans le cas d'une exécution en présence d'un attaquant actif disposant d'un nombre arbitraire de sessions. Bien sûr, la transformation ne peut pas préserver *toutes* les propriétés imaginables. Ainsi, le protocole initial pouvait éventuellement garantir l'anonymat des participants, ce qui est perdu par l'utilisation du chiffrement à clefs publiques.

Nous avons identifié une classe $\mathcal{L}'$ de formules logiques (proche de la logique définie page 33 du chapitre 3) qui permet de spécifier des propriétés de sécurité préservées par notre transformation. La classe $\mathcal{L}'$ permet en particulier d'énoncer des propriétés telles que la confidentialité ou de nombreuses variantes de l'authentification. Informellement, nous avons obtenu le résultat suivant *[CWZ07b]* :

> *Si le protocole P satisfait une formule ϕ de $\mathcal{L}'$ lorsqu'il est exécuté au plus une fois sans la présence d'un attaquant, alors le protocole $t(P)$ obtenu après transformation satisfait ϕ même lorsqu'il est exécuté un nombre arbitraire de fois en présence d'un attaquant actif.*

En particulier, notre transformation protège le protocole Wide Mouthed Frog contre l'attaque par rejeu mentionnée en introduction. L'identifiant de sessions assure en effet la fraîcheur des données.

5.1.4 Discussion

L'inconvénient de notre transformation est qu'elle utilise une infrastructure assez lourde (comme l'envoi de tous les nonces à tous les participants dès le début du protocole ou l'utilisation de chiffrement à clefs publiques). Notre procédure n'est donc pas adaptée pour des protocoles spécialement conçus pour utiliser peu de ressources. Notre transformation doit plutôt être vue comme un compilateur par défaut, lorsqu'un utilisateur souhaite obtenir un protocole sûr sans de soucier des primitives cryptographiques. Ainsi, le protocole de départ n'a besoin de contenir aucune primitive cryptographique. Il suffit à l'utilisateur de spécifier dans le protocole initial la conversation normale souhaitée, sans se préoccuper d'attaques éventuelles.

D'autre part, il est intéressant de remarquer que notre transformation garantit la sécurité du protocole transformé non seulement dans le cadre d'un modèle symbolique mais également dans le cadre d'un modèle cryptographique, contre n'importe quel attaquant polynomial. Les primitives cryptographiques utilisées permettent en effet d'utiliser le résultat de transfert *[CW05]* présenté au chapitre 8.2. Aussi, le protocole obtenu après transformation garantit les propriétés de sécurité initiales, même contre n'importe quelle machine de Turing probabiliste polynomiale.

5.2 Combinaison de protocoles

Même lorsqu'un protocole est sûr pour un nombre non borné de sessions et contre un adversaire actif, plus aucune sécurité n'est garantie si le protocole est exécuté en même temps que d'autres protocoles partageant certaines de ses clefs, comme des clefs publiques par exemple. L'exemple naïf ci-dessous illustre bien le problème de la composition de protocoles. Considérons les deux protocoles suivants :

$$P_1 : \quad A \to B : \ \{s\}_{\mathsf{pub}(B)} \qquad\qquad P_2 : \quad \begin{array}{l} A \to B : \ \{N_a\}_{\mathsf{pub}(B)} \\ B \to A : \ N_a \end{array}$$

Dans le protocole P_1, l'agent A envoie un secret s chiffré par la clef publique de B. Dans le protocole P_2, l'agent A envoie un nonce frais N_a à B, chiffré par la clef publique de B. L'agent B accuse réception du message en renvoyant le nonce N_a en clair. Le protocole P_1 exécuté seul garantit sans peine la confidentialité de la donnée s mais peut être facilement attaqué si le protocole P_2 est exécuté. En effet, l'intrus peut se servir de P_2 comme un oracle de déchiffrement. D'autres exemples plus réalistes d'interaction sont détaillés dans [KSW97].

5.2.1 Ajouter des étiquettes suffit pour la combinaison sûre de protocoles

Nous avons montré *[CDD07a, CD09c]* que pour préserver la sécurité des protocoles, il suffit d'étiqueter les protocoles. Notre résultat est valide pour un fragment important (noté PS-LTL$^+$) de la logique PS-LTL [Cor06], qui permet d'exprimer par exemple la confidentialité d'une donnée et de nombreuses variantes de l'authentification.

Plus précisément, nous avons montré que le résultat suivant

> *Si un protocole P satisfait une formule ϕ de PS-LTL$^+$, alors pour tout protocole Q, la composition parallèle de P et Q satisfait ϕ*
>
> $$P \models \phi \ \Rightarrow \ \forall Q, \quad P \mid Q \models \phi$$
>
> *à condition que les deux conditions suivantes soient satisfaites :*

1. *Les données communes à P et Q sont, soit publiques (e.g. constantes, noms d'agents ou clefs publiques), soit apparaissent uniquement en position de clef.*
2. *Pour tout sous-terme u de P, pour tout sous-terme v de Q tels que u et v commencent par un symbole de chiffrement, alors u et v ne sont pas unifiables.*

La première condition est satisfaite par exemple quand P et Q partagent des données publiques ainsi que des clefs long-termes qui ne sont utilisées que pour chiffrer et déchiffrer les messages. La seconde condition est facilement assurée par l'ajout d'étiquettes à l'intérieur de chaque chiffrement. Il suffit par exemple d'ajouter un identifiant, comme le nom du protocole, à l'intérieur de chaque message chiffré. Cet identifiant n'a bien sûr pas besoin d'être renouvelé à chaque session, une constante par protocole suffit.

Ainsi, la version étiquetée des protocoles P_1 et P_2 décrits ci-dessus est :

$$P_1' : \quad A \to B : \ \{1, s\}_{\mathsf{pub}(B)} \qquad\qquad P_2' : \quad \begin{array}{ll} A \to B : & \{2, N_a\}_{\mathsf{pub}(B)} \\ B \to A : & N_a \end{array}$$

Notre résultat de composition assure que P_1' peut être exécuté en présence de P_2', sans compromettre la confidentialité de s.

L'ajout d'étiquettes a été suggéré par Guttman and Thayer [GT00]. Ils montrent que deux protocoles peuvent être exécutés ensemble dès que les protocoles sont "indépendants". Leur condition d'indépendance requière que les messages chiffrés envoyés sont différents. L'hypothèse porte non sur la spécification du protocole mais sur toutes les exécutions possibles. Dans notre exemple du protocole P_2', l'agent B peut accepter un message de la forme $\{2, \{1, m\}_k\}_{\mathsf{pub}(B)}$. Un tel protocole ne satisfait donc pas la notion d'indépendance de [GT00] alors qu'il ne compromet pas la sécurité de P_1'.

5.2.2 Applications

Partager des clefs entre différents protocoles est souvent considéré comme dangereux et de ce fait, des architectures de clefs différentes sont en général utilisées pour les protocoles. Pourtant, partager les clefs permettrait à la fois d'économiser de la mémoire (pour stocker les clefs) et du temps (pour les générer). D'autre part, il existe déjà différentes situations où les clefs sont partagées :

- Plusieurs versions d'un même protocole peuvent être utilisées simultanément (tous les utilisateurs n'ont pas la dernière version à jour). Dans ce cas et pour assurer la compatibilité des différentes versions, les mêmes clefs sont utilisées dans chacune des versions, ce qui peut conduire à des failles.
- Deux protocoles sont couramment utilisés pour chiffrer le courrier électronique et utilisent la même clef publique : le protocole PGP (Pretty Good Privacy) et sa version publique OpenPGP. Le protocole PGP contient également des sous-protocoles tels que la signature électronique de messages, qui font appel à la même infrastructure à clefs publiques.
- Dans le contexte un peu différent des interfaces de programmation présentées au chapitre 4, partie 4.3, J. Clulow [Clu03a] a découvert une attaque lorsque le protocole de vérification des codes secrets (VISA PIN verification values) et l'interface IBM CCA API partagent la même clef de vérification.

5.3 Perspectives

Le développement des protocoles est de plus en plus complexe. Si la vérification d'un protocole isolé est souvent possible à l'aide des outils logiciels existants (comme par exemple ProVerif [Bla01, Bla05], Scyther [Cre08b, Cre08a] ou Avispa [ABB+05]), l'analyse d'un protocole comportant plus d'une dizaine d'étapes dépasse souvent la capacité des outils. Il est donc difficile d'analyser d'un seul tenant un protocole faisant appel à des sous-protocoles ou un groupe de protocoles amenés à être installés sur une même machine hôte. Il apparaît donc un besoin important de méthodologies pour développer les protocoles de façon modulaire, de manière à ne faire appel aux outils que pour chacun des composants. Nous développerons plus en détail au chapitre 10 différentes directions de recherche plus générales autour du thème du développement modulaire de protocoles. Nous proposons ici quelques développements directs des résultats présentés dans ce chapitre.

Comme nous l'avons indiqué dans la partie 5.1.4, notre compilateur requière une infrastructure relativement lourde (diffusion des nonces lors de la première étape, utilisation des clefs publiques). Plusieurs pistes pourraient en améliorer les performances. Tout d'abord, au lieu de mettre en place une première étape où chacun envoie un nonce frais à tous les autres participants, il semble possible de ne pas augmenter le nombre de messages échangés en construisant le numéro de sessions au fur et à mesure que les agents sont impliqués dans le protocole. Cela éviterait par exemple d'envoyer des requêtes à un serveur qui n'interviendrait finalement pas dans la session. Pour alléger l'utilisation du chiffrement à clef publique qui demande un temps de calcul important, nous envisageons d'introduire du chiffrement à clef symétrique à l'aide d'une clef de sessions établie, elle aussi, au fur et à mesure des communications.

D'autre part, nous pensons explorer plus largement quelles sont les propriétés garanties par notre transformation et éventuellement modifier la transformation de manière à couvrir plus de propriétés. Ainsi, notre transformation a été reprise et allégée par Myrto Arapanis, Stéphanie Delaune et Steve Kremer [ADK08] pour montrer qu'il possible d'obtenir un protocole sûr pour un nombre arbritraire de sessions, à partir d'un protocole sûr pour une seule session, en ajoutant le même identifiant de sessions que celui que nous avons proposé. Les hypothèses requises sont donc un peu plus forte (puisque le protocole doit être sûr pour une session contre un attaquant *actif*) mais la transformation est plus légère puisqu'il n'est pas nécessaire de chiffrer et de signer à nouveau les messages.

Nous avons montré que l'ajout d'étiquettes permet la composition sûre de protocoles pour des propriétés de type secret et authentification. Ainsi, notre technique a été preise par Stéphanie Delaune, Steve Kremer et Marc Ryan pour montrer qu'il était possible de composer l'équivalence statique pour les protocoles avec mots de passe [DKR08]. Stefan Ciobâcă [Cio08] a également montré qu'il était possible de composer l'équivalence statique pour des protocoles avec les primitives classiques de chiffrement et concaténation. Il semble à nouveau tout à fait possible d'étendre ce résultat à des propriétés de sécurité formulées à l'aide d'équivalence observationelle (notée $\sim_o$ et définie au chapitre 7). Le résultat recherché serait alors de la forme :

$$\nu\tilde{k}P_1 \sim_o \nu\tilde{k}P_2 \;\Rightarrow\; \nu\tilde{k}(P_1 \mid Q) \sim_o \nu\tilde{k}(P_2 \mid Q)$$

Un premier pas a été esquissé par Joshua Guttman [Gut09] mais les hypothèses proposées n'ont pas la simplicité des étiquettes que nous proposons et semblent difficiles à vérifier.

Deuxième partie

Analyse de propriétés d'équivalence

CHAPITRE 6

Analyse de la connaissance de l'adversaire

Ce chapitre est consacré à l'analyse de la *connaissance* de l'intrus. Avant même de considérer l'interaction d'un attaquant avec un protocole, il est nécessaire de d'évaluer la connaissance qu'un intrus obtient à partir d'un ensemble de message. Nous avons déjà évoqué la capacité de *déduction* de l'intrus : à partir d'un ensemble de messages S, quels sont les messages déductibles (constructibles) de S ? Cette relation, notée $\vdash$, a été définie pour différentes primitives cryptographiques aux chapitres précédents. Cette notion ne reflète cependant pas toute la connaissance accessible par un intrus. Prenons l'exemple classique de l'expression d'un vote et supposons qu'un votant envoie le message $\{i\}_{\mathsf{pub}(S)}$ à un serveur où $i \in \{0,1\}$ représente la valeur de son vote. Un attaquant peut alors connaître la valeur du vote, non pas en déchiffrant le message, mais en construisant les messages $\{0\}_{\mathsf{pub}(S)}$ et $\{1\}_{\mathsf{pub}(S)}$ et en comparant les deux chiffrés au message envoyé (à condition que le chiffrement soit déterministe).

Pour refléter la capacité de comparaison d'un intrus, Martín Abadi et Cédric Fournet ont introduit [AF01] la notion d'*équivalence statique* notée $\approx$. Deux ensembles de messages sont dits équivalents statiquement si un attaquant ne peut pas les différencier, c'est-à-dire s'il ne peut pas construire de test qui les différencie. Cette notion est proche de la notion d'*indistinguabilité* très utilisée en cryptographie [GM84]. Nous établirons une comparaison formelle entre ces deux notions au chapitre 9.1.

Dans ce chapitre, nous explorons la décidabilité de l'équivalence statique, pour des primitives cryptographiques variées.

6.1 Définitions

Nous considérons une signature $\mathcal{F}$, un ensemble de variables $\mathcal{X}$ et un ensemble de noms $\mathcal{N}$, munis d'une théorie équationnelle E. Les séquences de messages $M_1, \ldots, M_l$ sont organisées en *structure* (« frame » en anglais) de la forme $\phi = \nu\tilde{n}.\sigma$ où $\tilde{n}$ est un ensemble fini de noms dits *restreints* (initialement inconnus de l'intrus) et σ est une substitution de la forme

$$\sigma = \{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}.$$

Les variables x_i peuvent être vues comme des pointeurs sur les messages M_i. L'opérateur ν est l'opérateur de restriction du pi calcul [Mil99], que l'on retrouvera au chapitre 7. La taille d'une structure $\phi = \nu\tilde{n}.\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ est $|\phi| = \sum_{i=1}^{l} |M_i|$. Les noms $\tilde{n}$ sont liés dans ϕ et peuvent être renommés.

6.1.1 Déduction

Étant donnée une théorie équationnelle E, la notion de déduction est définie de manière canonique par les règles suivantes :

$$\frac{}{\nu\tilde{n}.\sigma \vdash_E M} \text{ si } \exists x \in \mathsf{dom}(\sigma) \text{ tq. } x\sigma = M \qquad \frac{}{\nu\tilde{n}.\sigma \vdash_E s} \; s \notin \tilde{n}$$

$$\frac{\phi \vdash_E M_1 \quad \cdots \quad \phi \vdash_E M_k}{\phi \vdash_E f(M_1, \ldots, M_k)} \; f \in \Sigma \qquad \frac{\phi \vdash_E M \quad M =_E M'}{\phi \vdash_E M'}$$

Intuitivement, les messages déductibles de ϕ sont les messages de ϕ ainsi que les noms non restreints de ϕ, clos par égalité dans E et par application de symboles fonctionnels.

La proposition suivante permet d'introduire la notion de *recette* associée à un terme déductible.

Proposition 6.1 (*[AC04a]*) *Soit M un terme clos et $\nu\tilde{n}.\sigma$ une structure. Alors $\nu\tilde{n}.\sigma \vdash_E M$ si et seulement si il existe un terme ζ tel que $\mathsf{noms}(\zeta) \cap \tilde{n} = \emptyset$ and $\zeta\sigma =_E M$. Le terme ζ est appelé* recette *associée à M.*

Un terme déductible peut admettre plusieurs recettes.

Considérons la théorie E_{enc} définie page 25 ainsi que la structure $\phi \stackrel{\mathsf{def}}{=} \nu\{k, s\}.\{\mathsf{enc}(s, k)/x, k/y\}$. Alors $\phi \vdash_{E_{\mathsf{enc}}} k$ et $\phi \vdash_{E_{\mathsf{enc}}} s$. De plus, une recette pour k est y puisque $k =_{E_{\mathsf{enc}}} y\phi$ et une recette pour s est $\mathsf{dec}(x, y)$ puisque $s =_{E_{\mathsf{enc}}} \mathsf{dec}(x, y)\phi$.

6.1.2 Équivalence statique

Deux termes M et N sont égaux dans la structure φ pour la théorie équationnelle E, ce qui sera noté $(M =_E N)\varphi$, si et seulement si $\varphi = \nu\tilde{n}.\sigma$, $M\sigma =_E N\sigma$ et $\{\tilde{n}\} \cap (\mathsf{noms}(M) \cup \mathsf{noms}(N)) = \emptyset$ pour un choix de $\tilde{n}$ et d'une substitution σ.

Deux structures φ et ψ sont *statiquement équivalentes*, noté $\varphi \approx_E \psi$, si $\mathsf{dom}(\varphi) = \mathsf{dom}(\psi)$ et si, pour tous termes M et N, la propriété suivante est vérifiée :

$$(M =_E N)\varphi \;\Leftrightarrow (M =_E N)\psi.$$

Intuitivement, φ et ψ sont statiquement équivalentes si elles vérifient les mêmes égalités.

Considérons par exemple la structure $\phi_1 \stackrel{\mathsf{def}}{=} \nu k.\{\mathsf{enc}(0, k)/x, k/y\}$ correspondant au cas où un votant vote 0 à l'aide de la clef k ainsi que la structure $\phi_2 \stackrel{\mathsf{def}}{=} \nu k.\{\mathsf{enc}(1, k)/x, k/y\}$, correspondant au cas où un votant vote 1 à l'aide de la clef k. Les valeurs 0 et 1 sont des symboles constants donc connus de l'intrus. ϕ_1 satisfait l'égalité $(\mathsf{dec}(x, y) =_{E_{\mathsf{enc}}} 0)\phi_1$ ce qui n'est pas le cas de ϕ_2 : $(\mathsf{dec}(x, y) \neq_{E_{\mathsf{enc}}} 0)\phi_2$. On en déduit $\phi_1 \not\approx_{E_{\mathsf{enc}}} \phi_2$ alors que $\nu k.\{\mathsf{enc}(0, k)/x\} \approx_{E_{\mathsf{enc}}} \nu k.\{\mathsf{enc}(1, k)/x\}$.

6.1.3 Comparaison des deux notions

Intuitivement, l'équivalence statique est plus forte que la déduction. Ceci est vrai dès que la théorie équationnelle contient un symbole fonctionnel libre.

Proposition 6.2 (*[AC04a, AC06]*) *Soit E une théorie équationnelle associée à une signature $\mathcal{F}$. Soit $\mathcal{F}' \stackrel{\text{def}}{=} \mathcal{F} \uplus \{h\}$, où h est un symbole unaire. Soit E' la plus petite théorie équationnelle étendant E aux termes de $\mathcal{F}'$. Soit $\phi = \nu\tilde{n}.\{^{x_1}/_{M_1}, \ldots, ^{x_l}/_{M_l}\}$ une structure sur $\mathcal{F}$, M un terme clos sur $\mathcal{F}$ et k un nom frais. Alors $\phi \vdash_E M$ si et seulement si*

$$\nu\tilde{n}.\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}, ^{h(M)}/_{x_{l+1}}\} \not\approx_{E'} \nu(\tilde{n}. \cup \{k\})\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}, ^{k}/_{x_{l+1}}\}$$

Nous pouvons en déduire que si $\approx_{E'}$ est décidable, alors $\vdash_E$ est aussi décidable (avec au plus la même complexité). Cependant, l'existence d'un symbole libre (ou d'une construction jouant le même rôle, comme le chiffrement par exemple) est important pour la réduction de la déduction à l'équivalence statique. Ainsi, si on considère uniquement la théorie AC pure E_{AC} (définie page 62) alors $\approx_{E_{\mathsf{AC}}}$ est décidable en temps polynomial alors que $\vdash_{E_{\mathsf{AC}}}$ est NP-complet, ce qui montre que la réduction proposée à la proposition 6.2 n'est pas toujours possible.

À l'inverse, il est possible de construire une théorie équationnelle E telle que $\vdash_E$ est décidable alors que $\approx_E$ ne l'est pas. Une première construction avait été proposée dans *[AC04a]* avec une esquisse de preuve, une preuve complète a été proposée par Borgström [Bor05].

La suite de ce chapitre est consacrée à la caractérisation de théories équationnelles E pour lesquelles les relations $\vdash_E$ et $\approx_E$ sont décidables.

6.2 Cas des théories sous-termes

Une première classe de théories équationnelles pour lesquelles déduction et équivalence statique sont décidables est la classe des théories sous-termes convergentes.

Définition 6.1 (Théories sous-termes) *Une théorie E est dite* sous-terme *si elle peut être définie par un ensemble fini d'équations de la forme $M = N$ où N est un sous-terme de M ou une constante.*

Une théorie E est sous-terme convergente si E est sous-terme et convergente.

Ainsi, la théorie E_{enc} est sous-terme convergente. C'est également le cas des trois théories définies ci-dessous :

$$\begin{array}{ll} E_{\mathsf{inv}} : & \{I(I(x)) = x, I(x) \times x = 1, x \times I(x) = 1\} \\ E_{\mathsf{idem}} : & \{h(h(x)) = h(x)\} \\ E_{\mathsf{sym}} : & \{\mathsf{enc}(\mathsf{enc}(x, y), y) = x\} \end{array}$$

La théorie E_{inv} modélise la fonction inverse dans les groupes par exemple. La théorie E_{idem} représente une fonction de hachage idempotente sur des entrées de petite taille (puisque le hachage d'un message haché $h(m)$ produit à nouveau $h(m)$). La théorie E_{sym} représente une fonction de chiffrement qui permet également de déchiffrer.

Théorème 6.1 (*[AC04a, AC06]*) *Soit E une théorie sous-terme convergente. Les deux problèmes $\phi \vdash M$ et $\phi \approx \phi'$ sont décidables en temps polynomial en la taille de ϕ, ϕ' et $|M|$.*

Ce résultat a été étendu par Mathieu Baudet [Bau05, Bau07] au cas des théories convergentes, définies par un ensemble fini d'équations de la forme $M = N$ où N est un sous-terme de M ou un *terme constant*.

6.3 Extensions et implémentation

En utilisant les techniques de preuves développées pour le théorème 6.1, nous avons mis au point des conditions suffisantes pour la décidabilité de la déduction et de l'équivalence statique. Ce résultat a été généralisé par Mathieu Baudet [Bau05, Bau07] dans le cas actif, pour les théories sous-terme convergentes. En collaboration avec Mathieu Baudet, nous avons ensuite revisité sa procédure pour proposer un algorithme efficace (et implémenté) qui permet de décider la déduction et de l'équivalence statique pour une large classe de théories équationnelles, incluant en particulier les théories sous-termes convergentes.

6.3.1 Conditions suffisantes pour des théories équationnelles AC-convergentes

Soit E une théorie équationnelle. Les symboles AC de E sont les symboles binaires $\oplus_1, \ldots, \oplus_k$ tels que les équations $x \oplus_i (y \oplus_i z) = (x \oplus_i y) \oplus_i z$ (associativité) et $x \oplus_i y = y \oplus_i x$ (commutativité) sont dans E. Nous écrivons qu'une théorie E est *AC-convergente* si elle est induite par un système de réécriture $\rightarrow$ terminant et confluent modulo AC, tel que $U =_E V$ si et seulement si l'intersection de leurs formes normales modulo AC est non vide. Si E ne contient pas de symbole AC alors on retrouve la notion habituelle de convergence.

L'idée générale pour montrer la décidabilité de l'équivalence statique est de se ramener à un ensemble de *petites équations*

$$\mathsf{Eq}(\phi) = \{(M = N) \mid (M =_E N)\phi \text{ et } |M|, |N| \leqslant c\}$$

où c est une constante bien choisie et où la notion de taille ne compte pas les symboles AC : $|t_1 \oplus t_2| = \max(|t_1|, |t_2|)$. L'ensemble des petites équations $\mathsf{Eq}(\phi)$ est fini si E ne contient pas de symbole AC mais peut être infini si E contient des symboles AC.

Nous avons montré *[AC05, AC06]* que la déduction et l'équivalence statique sont décidables pour une théorie E AC-convergente dès lors que :

1. E est localement stable : pour toute structure ϕ, on peut construire un ensemble $\mathsf{sat}(\phi)$, stable par application d'un petit contexte

   $$\forall t_1, \ldots, t_k \in \mathsf{sat}(\phi)\ \forall C \text{ tq. } |C| \leq c',\ C[t_1, \ldots, t_k] \rightarrow_{AC} C'[t'_1, \ldots, t'_n]$$

   pour des termes $t'_1, \ldots, t'_n \in \mathsf{sat}(\phi)$ et C' un (petit) contexte. La constante c' dépend de la taille de la théorie E.

2. E est localement décidable : savoir si ϕ' satisfait les équations de $\mathsf{Eq}(\phi)$ est décidable.

La condition 1 est en particulier vérifiée pour toutes les théories sous-terme convergentes. La condition 2 est toujours vérifiée pour les théories E sans symbole AC puisque $\mathsf{Eq}(\phi)$ est alors fini. Dans le cas de la théorie $E_\oplus$ du XOR avec un symbole AC $\oplus$, l'ensemble $\mathsf{Eq}(\phi)$ est également fini (modulo XOR). Dans le cas de la théorie E_{AC} AC pure, induite par les équations

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \quad \text{et} \quad x \oplus y = y \oplus x,$$

on peut montrer qu'on peut réduire $\mathsf{Eq}(\phi)$ à un ensemble fini en calculant le noyau d'un $\mathbb{Z}$-module [Sch86]. Cette approche sera détaillée et généralisée à la partie 6.4.

À l'aide de ce résultat, nous retrouvons la décidabilité de la déduction et de l'équivalence statique pour les théories sous-terme convergentes. Nous avons également pu montrer la décidabilité de la déduction et de l'équivalence statique pour d'autres théories non sous-terme

convergentes comme les théories $E_{\oplus}$, E_{AC} ou E_{blind} et E_{homo} définies ci-dessous.

$$E_{\mathsf{blind}} = E_{\mathsf{enc}} \cup \left\{ \begin{array}{rcl} \mathsf{open}(\mathsf{commit}(x,y),y) & = & x \\ \mathsf{getpk}(\mathsf{host}(x)) & = & x \\ \mathsf{checksign}(\mathsf{sign}(x,y),\mathsf{pk}(y)) & = & x \\ \mathsf{unblind}(\mathsf{blind}(x,y),y) & = & x \\ \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) & = & \mathsf{sign}(x,z) \end{array} \right\}$$

$$E_{\mathsf{homo}} = E_{\mathsf{enc}} \cup \left\{ \begin{array}{rcl} \mathsf{enc}(\mathsf{pair}(x,y),z) & = & \mathsf{pair}(\mathsf{enc}(x,z),\mathsf{enc}(y,z)) \\ \mathsf{dec}(\mathsf{pair}(x,y),z) & = & \mathsf{pair}(\mathsf{dec}(x,z),\mathsf{dec}(y,z)) \end{array} \right\}$$

La théorie E_{blind} a été introduite par S. Kremer and M. Ryan pour modéliser la signature en aveugle, primitive utilisée en particulier dans certains protocoles de vote électronique [KR05]. La théorie E_{homo} représente un schéma de chiffrement homomorphique comme le mode de chiffrement ECB (Encryption Chaining Block) par exemple, où chaque bloc est chiffré indépendemment des précédents.

D'autres exemples de théories localement stables et localement décidables (donc pour lesquelles la déduction et de l'équivalence statique sont décidables) ont été proposés dans *[AC06]*.

6.3.2 L'outil YAPA

Mathieu Baudet a étendu [Bau05, Bau07] le théorème 6.1 au cas actif, pour décider l'existence d'attaques par dictionnaire pour un nombre borné de sessions. On appelle « attaques par dictionnaire » les attaques consistant pour l'intrus à essayer par force brute les différentes valeurs d'un secret faible comme un mot de passe par exemple.

Nous avons revisité *[BCD09a]* l'algorithme proposé par Mathieu Baudet, dans le cas passif uniquement, de manière à pouvoir traiter une classe plus large de théories que les théories sous-termes (mais sans symbole AC). Le principe de la procédure consiste à saturer une structure ϕ en ajoutant les termes déductibles par application d'un petit contexte, en n'ajoutant que les termes qui n'étaient pas déjà constructibles à partir des termes courants. Les recettes associées aux termes déductibles sont calculées au vol. L'algorithme est correct et complet au sens où, si l'outil parvient à saturer la structure ϕ, alors

- un terme est déductible si et seulement si il est syntaxiquement déductible à partir de la structure saturée,
- une structure ϕ' est statiquement équivalente à ϕ si et seulement si elle vérifie les égalités entre recettes calculées lors de la saturation.

Par contre, il se peut que la saturation échoue ou ne termine pas. Nous avons montré que la procédure de saturation n'échoue jamais pour la classe des théories *convergentes en couche* comme les théories E_{blind} ou E_{homo} ou la théorie E_{pref}, définie ci-dessous.

$$E_{\mathsf{pref}} = E_{\mathsf{enc}} \cup \left\{ \begin{array}{rcl} \mathsf{pref}(\mathsf{enc}(\mathsf{pair}(x,y),z)) & = & \mathsf{enc}(x,z) \end{array} \right\}$$

Nous avons également proposé un critère pour assurer la terminaison de la procédure pour les théories convergentes en couche. Ce critère est en particulier satisfait par toutes les théories localement stables (sans symbole AC). Cet algorithme nous a permis de déduire la décidabilité de la déduction et de l'équivalence statique pour la théorie E_{pref} (décidabilité qui aurait aussi pu être établie à l'aide du résultat de *[AC05, AC06]*).

Cet algorithme a été implémenté dans l'outil YAPA[1] et fonctionne de manière efficace, comme l'illustre le tableau ci-dessous.

[1] Accessible sur la page `http://www.lsv.ens-cachan.fr/~baudet/yapa/`

Théorie équationnelle	E_{enc} $n = 10$	E_{enc} $n = 14$	E_{enc} $n = 18$	E_{enc} $n = 20$	E_{blind}	E_{pref}	E_{homo}
Temps d'exécution	< 1s	1,7s	30s	< 3min	< 1s	< 1s	< 1s

Dans le cas de la théorie équationnelle E_{enc}, nous avons testé YAPA sur les structures $\varphi_n = \{{}^{t_n^0}/_{x_1}, {}^{c_0}/_{x_2}, {}^{c_1}/_{x_3}\}$ et $\varphi'_n = \{{}^{t_n^1}/_{x_1}, {}^{c_0}/_{x_2}, {}^{c_1}/_{x_3}\}$, où $t_0^i = c_i$ et $t_{n+1}^i = \mathsf{pair}(\mathsf{enc}(t_n^i, k_n^i), k_n^i)$, $i \in \{0, 1\}$. Ces exemples permettent d'accroître exponentiellement en n la taille (non DAG) des tests permettant de distinguer φ_n et φ'_n tandis que la taille de φ_n et φ'_n croît linéairement. Les fichiers d'exemples utilisés pour les théories E_{blind}, E_{pref} et E_{homo} sont accessibles depuis la page de l'outil.

YAPA est le premier outil dédié à la décision de la déduction et l'équivalence statique. Il propose un algorithme de décision unifiée, fonctionnant pour la plupart des théories sans symbole AC. Le seul autre outil capable de décider l'équivalence statique est l'outil ProVerif [Bla01, Bla05], développé par Bruno Blanchet. Cet outil est conçu pour analyser des propriétés d'accessibilité et d'équivalence pour les protocoles cryptographiques, pour un nombre non borné de sessions. Il s'applique donc à un contexte plus large (et plus difficile) que YAPA. En retour, ProVerif est nettement moins efficace pour décider l'équivalence statique et échoue sur certaines théories comme E_{homo}.

6.4 Théories monoïdales

Les théories monoïdales, définies par W. Nutt [Nut90], regroupent de nombreuses théories avec symbole AC. Nous avons proposé *[CD07a]* une méthode générale pour réduire la décidabilité de la déduction et de l'équivalence statique à des problèmes algébriques mieux connus.

6.4.1 Définitions

Définition 6.2 (Théorie monoïdale) *Une théorie E sur la signature $\mathcal{F}$ est dite* monoïdale *si elle satisfait les trois propriétés suivantes :*

1. *La signature $\mathcal{F}$ contient un symbole binaire $+$ ainsi qu'un symbole constant 0 et tous les autres symboles fonctionnels de $\mathcal{F}$ sont unaires.*
2. *Le symbole $+$ est associatif et commutatif, avec pour unité 0,* i.e. *les équations $x+(y+z) = (x+y)+z$ (A), $x+y = y+x$ (C) et $x+0 = x$ (U) sont dans E.*
3. *Chaque symbole unaire $h \in \mathcal{F}$ est un endomorphisme pour $+$ et 0, i.e. $h(x+y) = h(x)+h(y)$ et $h(0) = 0$.*

Les théories ci-dessous sont monoïdales.
- La théorie ACU sur $\mathcal{F} = \{+, 0\}$ induite par les axiomes (A),(C) et (U).
- La théorie ACUI sur $\mathcal{F} = \{+, 0\}$ induite par les axiomes (A),(C), (U) et (I) : $x + x = x$ (Idempotence).
- La théorie ACUN (théorie du *ou exclusif*, notée $E_{\oplus}$ jusqu'ici) induite par les axiomes (A),(C), (U) et (N) : $x + x = 0$.
- La théorie AG (*groupes Abéliens*) sur $\mathcal{F} = \{+, -, 0\}$ induite par les axiomes (A),(C), (U) et $x + -(x) = 0$.

- Les théories ACUh, ACUIh, ACUNh sur $\mathcal{F} = \{+, h, 0\}$ et AGh sur $\mathcal{F} = \{+, -, h, 0\}$: ces théories correspondent aux théories précédentes, étendues par les lois d'homomorphisme : $h(x + y) = h(x) + h(y)$ et $h(0) = 0$.
- La théorie $\mathsf{AGh_1 \ldots h_n}$ sur $\mathcal{F} = \{+, -, h_1, \ldots, h_n, 0\}$, induite par les axiomes de AG, les lois d'homomorphisme pour chaque h_i et la commutativité des symboles unaires : $h_i(h_j(x)) = h_j(h_i(x))$ pour tout $1 \leqslant i, j \leqslant n$.

D'autres exemples de théories monoïdales sont présentés dans [Nut90].

Stéphanie Delaune a montré [Del06a] que le problème de la déduction pour la théorie ACU se réduit à la résolution linéaire d'équations dans $\mathbb{N}$ tandis que le problème de la déduction pour la théorie AGh se réduit à la résolution linéaire d'équations dans $\mathbb{Z}[h]$, l'anneau des polynômes à une indéterminée, à coefficients dans $\mathbb{Z}$. Nous avons vu à la partie 6.3.1, que la déduction et l'équivalence statique sont décidables pour les théories ACUN et AC.

Nous avons généralisé ces résultats en associant un semi-anneau à chaque théorie monoïdale. Cela nous a permis de réduire la décision de la déduction et de l'équivalence statique à des problèmes plus classiques en algèbre. Les théories monoïdales présentent une structure algébrique proche de celles des anneaux, à ceci près que certains éléments peuvent ne pas avoir d'inverse (pour la loi de « groupe »). Une telle structure est appelée *semi-anneau*.

Définition 6.3 (Semi-anneau) *Un* semi-anneau *est un ensemble $\mathcal{S}$ (appelé* univers *du semi-anneau) contenant les éléments 0 et 1 et muni de deux opérations binaires $+$ et $\cdot$ telles que $(\mathcal{S}, +, 0)$ est un monoïde commutatif, $(\mathcal{S}, \cdot, 1)$ est un monoïde, et telles que les propriétés suivantes sont vérifiées pour tous $\alpha, \beta, \gamma \in \mathcal{S}$:*

- $(\alpha + \beta) \cdot \gamma = \alpha \cdot \gamma + \beta \cdot \gamma$ *(distributivité à droite)*
- $\alpha \cdot (\beta + \gamma) = \alpha \cdot \beta + \alpha \cdot \gamma$ *(distributivité à gauche)*
- $0 \cdot \alpha = \alpha \cdot 0 = 0$ *(loi du zéro).*

Les opérations $+$ et $\cdot$ sont appelées respectivement *addition* et *multiplication* du semi-anneau $\mathcal{S}$. Les éléments 0 et 1 sont respectivement le *zéro* et l'*unité*.

W. Nutt a montré [Nut90] qu'on peut associer un semi-anneau $\mathcal{S}_E$ à toute théorie monoïdale E. Le semi-anneau $\mathcal{S}_E$ est construit de la manière suivante. Son univers est $T(\mathcal{F}, \{\mathbf{1}\})/E$, où $\mathbf{1}$ est une nouvelle constante libre ($\mathbf{1} \notin \mathcal{F}$). La constante 0 est le zéro de $\mathcal{S}_E$, le symbole $+$ en est l'addition. La multiplication est définie par $s \cdot t := s[\mathbf{1} \mapsto t]$. La constante $\mathbf{1}$ agit donc comme élément neutre de la multiplication.

Nous illustrons cette définition par quelques exemples.

1. Le semi-anneau $\mathcal{S}_{\mathsf{ACU}}$ est isomorphe à $\mathbb{N}$, le semi-anneau des entiers naturels.
2. Le semi-anneau $\mathcal{S}_{\mathsf{ACUN}}$ est isomorphe au corps $\mathbb{Z}/2\mathbb{Z}$.
3. Le semi-anneau $\mathcal{S}_{\mathsf{AGh}}$ à l'anneau commutatif $\mathbb{Z}[\mathsf{h}]$.

6.4.2 Réduction de la déduction et de l'équivalence statique

Considérons la théorie ACU ainsi que la structure

$$\phi = \nu n_1, n_2, n_3.\{^{3n_1+2n_2+3n_3}/_{x_1}, {}^{n_2+3n_3}/_{x_2}, {}^{3n_2+n_3}/_{x_3}, {}^{3n_1+n_2+4n_3}/_{x_4}\},$$

où la notation kn avec $k \in \mathbb{N}$ représente le terme $n + \cdots + n$ (k fois). On peut associer à ϕ la matrice M_ϕ où $i$ème ligne représente la décomposition de x_i sur les constantes n_1, n_2, n_3.

$$M_\phi = \begin{pmatrix} 3 & 2 & 3 \\ 0 & 1 & 3 \\ 0 & 3 & 1 \\ 3 & 1 & 4 \end{pmatrix}$$

Soit $t = 7n_1 + 3n_2 + 8n_3$. On associe à t le vecteur $U_t = \begin{pmatrix} 7 & 3 & 8 \end{pmatrix}$. On remarque facilement que t est déductible à partir de ϕ si et seulement si il existe $X \in \mathbb{N}^4$ tel que $XM_\phi = U_t$.

Cette réduction peut être généralisée à toutes les théories monoïdales.

Théorème 6.2 (*[CD07a]*) *Soit E une théorie monoïdale et $\mathcal{S}_E$ son semi-anneau associé. La déduction dans E se réduit en temps polynomial au problème suivant :*

Données *: Une matrice A de taille $\ell \times m$ et un vecteur de taille ℓ, à coefficients dans $\mathcal{S}_E$.*
Question *: Est-ce qu'il existe un vecteur X (à coefficients dans $\mathcal{S}_E$) tel que $X \cdot A = b$?*

Considérons maintenant la structure

$$\phi' = \nu n_1, n_2, n_3.\{^{n_1+2n_2+7n_3}/_{x_1}, {}^{n_1+5n_2}/_{x_2}, {}^{5n_2+8n_3}/_{x_3}, {}^{3n_1+2n_2+4n_3}/_{x_4}\}$$

On lui associe la matrice $M_{\phi'}$ définie ci-dessous.

$$M_{\phi'} = \begin{pmatrix} 1 & 2 & 7 \\ 1 & 5 & 0 \\ 0 & 5 & 8 \\ 3 & 2 & 4 \end{pmatrix}$$

Les structures ϕ et ϕ' sont statiquement équivalentes si et seulement si, pour tous termes M, N de $T(\{0, +, x_1, x_2, x_3, x_4\})$,

$$(M =_{\mathsf{ACU}} N)\phi \quad \Leftrightarrow \quad (M =_{\mathsf{ACU}} N)\phi'$$

Ceci est équivalent aux relations matricielles suivantes : pour tout $X \in \mathbb{Z}^4$, $XM_\phi = 0$ si et seulement si $XM_{\phi'} = 0$. C'est-à-dire, ϕ et ϕ' sont statiquement équivalentes si et seulement si les matrices M_ϕ et $M_{\phi'}$ ont même «'noyau ».

Cette réduction se généralise à nouveau à toutes les théories monoïdales.

Théorème 6.3 (*[CD07a]*) *Soit E une théorie monoïdale et $\mathcal{S}_E$ son semi-anneau associé. L'équivalence statique dans E se réduit en temps polynomial au problème suivant :*

Données *: Deux matrices A_1 et A_2 de taille $\ell \times m$ et à coefficients dans $\mathcal{S}_E$.*
Question *: Est-ce que l'égalité suivante est vérifiée ?*

$$\{(X, Y) \in \mathcal{S}_E^\ell \times \mathcal{S}_E^\ell \mid X \cdot A_1 = Y \cdot A_1\} = \{(X, Y) \in \mathcal{S}_E^\ell \times \mathcal{S}_E^\ell \mid X \cdot A_2 = Y \cdot A_2\}$$

Considérons la signature $\mathcal{F}_1 = \{+, 0, -, \mathsf{h}_1, \mathsf{h}_2\}$ et la théorie E_1 induite par les axiomes de AG, par l'équation $\mathsf{h}_1(\mathsf{h}_2(x)) = \mathsf{h}_2(\mathsf{h}_1(x))$, les lois d'homomorphisme

$$\begin{array}{rclcrcl} \mathsf{h}_1(x+y) & = & \mathsf{h}_1(x) + \mathsf{h}_1(y) & \qquad & \mathsf{h}_1(0) & = & 0 \\ \mathsf{h}_2(x+y) & = & \mathsf{h}_2(x) + \mathsf{h}_2(y) & & \mathsf{h}_2(0) & = & 0 \end{array}$$

ainsi que les équations

$$\begin{aligned} \mathsf{h}_1(\mathsf{h}_1(\mathsf{h}_2(x))) + \mathsf{h}_2(\mathsf{h}_2(x)) &= 0 \\ \mathsf{h}_1(x) + \mathsf{h}_1(\mathsf{h}_2(\mathsf{h}_2(x))) &= 0 \end{aligned}$$

La théorie E_1 est une théorie monoïdale et son semi-anneau associé $\mathcal{S}_{E_1}$ est isomorphe à $\mathbb{Z}[\mathsf{h}_1, \mathsf{h}_2]/_{(\mathsf{h}_1^2\mathsf{h}_2+\mathsf{h}_2^2, \mathsf{h}_1+\mathsf{h}_1\mathsf{h}_2^2)}$, *i.e.* l'anneau $\mathbb{Z}[\mathsf{h}]$ quotienté par l'idéal engendré par les polynômes $\mathsf{h}_1^2\mathsf{h}_2 + \mathsf{h}_2^2$ et $\mathsf{h}_1 + \mathsf{h}_1\mathsf{h}_2^2$. Les théorèmes 6.2 et 6.3 montrent que décider la déduction et l'équivalence statique revient à résoudre des systèmes linéaires dans $\mathbb{Z}[\mathsf{h}_1, \mathsf{h}_2]/_{(\mathsf{h}_1^2\mathsf{h}_2+\mathsf{h}_2^2, \mathsf{h}_1+\mathsf{h}_1\mathsf{h}_2^2)}$. Ce type de systèmes peut en général être résolu en utilisant les bases de Gröbner [Eis99] et des implémentations de ce type de résolution sont disponibles (comme dans le logiciel libre Sage[2]). La décision de la déduction et de l'équivalence statique peut donc être implémentée facilement à l'aide de ces outils.

6.5 Théories pour les protocoles de vote

Les protocoles de vote électronique utilisent souvent des primitives cryptographiques non standard comme la signature en aveugle décrite dans la partie 6.3.1. Nous nous sommes intéressés à deux exemples particuliers de théories équationnelles pertinentes pour les protocoles de vote, tirées de [DKR09b]. Nous décrivons ci-dessous ces deux théories puis nous en montrons la décidabilité.

6.5.1 Théorie pour le rechiffrement et preuve à vérificateur désigné

Le protocole de Lee *et al.* [LBD+03] fait appel à une primitive de rechiffrement $\mathsf{rencrypt}$ qui permet de modifier l'aléa du message chiffré

$$\mathsf{rencrypt}(\mathsf{enca}(x, \mathsf{pub}(y), z), w) = \mathsf{enca}(x, \mathsf{pub}(y), f_0(z, w)) \tag{6.1}$$

Le terme $\mathsf{enca}(m, \mathsf{pub}(a), r)$ représente le chiffrement à clefs publiques du message m par la clef $\mathsf{pub}(a)$. Le terme r représente l'aléa utilisé dans le chiffrement. Il permet de distinguer deux chiffrements d'un même message. Le symbole f_0 représente le fait que l'aléa est modifié par le rechiffrement.

Le protocole de Lee *et al.* fait également appel à une primitive de « preuve à vérificateur désigné ». Un agent peut produire la preuve $\mathsf{dvp}(x, y, z, \mathsf{pub}(w))$ que deux messages chiffrés x et y contiennent le même plaintext. Cette preuve n'est vérifiable que par l'agent w :

$$\mathsf{checkdvp}(\mathsf{dvp}(x, \mathsf{rencrypt}(x, y), y, \mathsf{pub}(z)), x, \mathsf{rencrypt}(x, y), \mathsf{pub}(z)) = \mathsf{ok} \tag{6.2}$$

$$\mathsf{checkdvp}(\mathsf{dvp}(x, y, z, w), x, y, \mathsf{pub}(w)) = \mathsf{ok} \tag{6.3}$$

La théorie E_{DVP}, définie dans [DKR09b], est formée des équations 6.1, 6.3 et 6.2 ainsi que des trois équations plus classiques ci-dessous :

$$\mathsf{checksign}(\mathsf{sign}(x, y), \mathsf{pub}(y)) = x \tag{6.4}$$

$$\mathsf{dec}(\mathsf{enca}(x, \mathsf{pub}(y), z), y) = x \tag{6.5}$$

$$\mathsf{getpk}(\mathsf{host}(x)) = x \tag{6.6}$$

La dernière équation permet à un agent d'obtenir la clef publique d'un autre agent.

[2] `http://www.sagemath.org/`

6.5.2 Théorie pour le « Trapdoor bit-commitment »

Le protocole de vote proposé par Fujioka, Okamoto et Ohta [FOO92] utilise une fonction d'engagement (commitment) $\mathsf{tdcommit}(x, y, z)$. Le premier argument de $\mathsf{tdcommit}$ est la valeur sur laquelle l'agent s'engage, le deuxième argument est la « trappe » qui permet d'ouvrir l'engagement :

$$\mathsf{open}(\mathsf{tdcommit}(x, y, z), y) = x \tag{6.7}$$

Cette fonction a la particularité de ne pas constituer un véritable engagement : le votant peut choisir la valeur de la trappe de manière à ouvrir l'engagement de la manière qu'il souhaite :

$$\mathsf{tdcommit}(x, f_1(y, z, w, x), w) = \mathsf{tdcommit}(y, z, w) \tag{6.8}$$

La théorie E_{Trap}, définie dans [DKR09b], est formée des équations 6.7, 6.8 et 6.6 ainsi que des trois équations ci-dessous pour la signature en aveugle.

$$\begin{aligned} \mathsf{checksign}(\mathsf{sign}(x, y), \mathsf{pub}(y)) &= x \\ \mathsf{unblind}(\mathsf{blind}(x, y), y) &= x \\ \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x, y), z), y) &= \mathsf{sign}(x, z) \end{aligned}$$

6.5.3 Décidabilité

Les théories E_{DVP} et E_{Trap} peuvent être complétées de manière à être convergentes. Cependant, aucun des résultats présentés dans les parties précédentes ne s'applique. En adaptant la technique développée à la partie 6.3.1, nous avons montré la décidabilité de la déduction et de l'équivalence statique pour ces deux théories.

Théorème 6.4 (*[BBRC09]*) *La déduction et l'équivalence statique sont décidables en temps polynomial pour les théories E_{DVP} et E_{Trap}.*

6.6 Composition

Chacun des résultats de décidabilité présentés dans les parties précédentes n'est valide que pour des théories équationnelles particulières ou pour des classes de théories particulières. Nous avons montré *[ACD07a]* qu'il est facile de composer les résultats de décidabilité dès lors que les théories sont disjointes.

Étant donnés une signature $\mathcal{F}$ et un ensemble E d'équations sur $\mathcal{F}$, on appelle théorie équationnelle $(\mathcal{F}, E)$ la relation d'équivalence induite sur $T(\mathcal{F}, \mathcal{X}, \mathcal{N})$ par les équations de E.

Théorème 6.5 (*[ACD07a]*) *Soient $(\mathcal{F}_1, E_1)$ et $(\mathcal{F}_2, E_2)$ deux théories équationnelles telles que $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$. Si la déduction et l'équivalence statique sont décidables pour $(\mathcal{F}_1, E_1)$ et pour $(\mathcal{F}_2, E_2)$, alors la déduction et l'équivalence statique sont décidables $(\mathcal{F}_1 \cup \mathcal{F}_2, E_1 \cup E_2)$.*

Ce résultat nous permet donc de combiner tous les résultats précédents, dès lors que les théories sont disjointes (mais peuvent partager les noms libres de $\mathcal{N}$).

Théorie	Déduction	Équivalence statique
sous-terme convergente Ex : $E_{\mathsf{enc}}, E_{\mathsf{inv}}, E_{\mathsf{idem}}, E_{\mathsf{sym}}$	PTIME *[AC06]*	
$E_{\mathsf{blind}}, E_{\mathsf{homo}}, E_{\mathsf{pref}}$	décidable *[AC06, BCD09a]*	
$E_{\mathsf{DVP}}, E_{\mathsf{Trap}}$	PTIME *[BBRC09]*	
ACU, E_{AC}	NP-complet	PTIME *[AC06, CD07a]*
ACUI	décidable [DLLT06, DLLT08]	décidable *[CD07a]*
ACUN ($E_{\oplus}$)	PTIME [CKRT03]	PTIME *[AC06, CD07a]*
AG	PTIME [CKR$^+$03a]	PTIME *[CD07a]*
ACUh	NP-complet [LLT05]	décidable *[CD07a]*
ACUIh	décidable *[CD07a]*	indéterminé
ACUNh	PTIME [Del06a]	décidable *[CD07a]*
AGh	PTIME [Del06a]	décidable *[CD07a]*
$\mathsf{AGh}_1 \ldots \mathsf{h}_n$	décidable *[CD07a]*	décidable *[CD07a]*
sous-terme conv. $\uplus$ $E_{\oplus}$	PTIME *[ACD07a]*	
sous-terme conv. $\uplus$ AG	PTIME *[ACD07a]*	

D'après le résultat de composition *[ACD07a]*, la déduction et l'équivalence statique sont également décidables pour l'union de toutes théories disjointes mentionnées dans ce tableau.

FIGURE 6.1 - *Résultats de décidabilité pour la déduction et l'équivalence statique.*

6.7 Bilan et perspectives

Les différents résultats de décidabilité décrits au cours de ce chapitre sont résumés à la figure 6.1. Il est probable que de nouvelles théories seront à étudier, en fonction des primitives utilisées par les protocoles cryptographiques.

L'outil YAPA propose la première implémentation dédiée à l'équivalence statique. Pour le moment, il ne permet pas de traiter les théories avec symbole AC. Nous avons montré que les théories monoïdales se ramènent à des problèmes algébriques pour lesquels il existe (souvent) des algorithmes déjà implémentés. Nous avons également vu (partie 6.6) qu'il est possible de décomposer une théorie en plusieurs sous-théories disjointes et de faire appel à des algorithmes différents pour chacune des théories. Il semble donc tout à fait faisable de combiner l'outil YAPA à un outil pour les théories monoïdales faisant appel au logiciel Sage par exemple. Appliquer directement le résultat de combinaison *[ACD07a]* posera peut-être des problèmes d'efficacité. Pour obtenir une implémentation efficace, il faudra probablement revisiter le résultat de combinaison de manière à l'intégrer directement à l'algorithme développé pour YAPA.

D'autre part, nous avons vu que la procédure de saturation implémentée dans YAPA peut

échouer. Il semble possible d'améliorer la procédure pour éliminer des cas d'échec et permettre à YAPA de saturer avec succès plus souvent. En particulier, l'outil YAPA ne permet pas pour le moment de traiter les théories E_{DVP} et E_{Trap} définies dans les protocoles de vote. Tenter d'étendre l'algorithme de YAPA à (au moins) ces deux théories paraît un bon objectif.

La décidabilité de la déduction et de l'équivalence statique ne sont qu'une brique de base pour analyser les protocoles contre un intrus actif. Une suite logique du travail présenté dans ce chapitre consiste à étendre les résultats de décidabilité au cas actif, au moins pour un nombre borné de sessions. Mathieu Baudet [Bau05, Bau07] a ainsi obtenu une procédure de décision pour la classe des théories sous-terme convergentes. Des procédures de décision ont également été obtenues pour le cas particulier du XOR [CKRT03] ainsi que pour les groupes abéliens [CKR+03a]. Par contre, aucun résultat n'existe à l'heure actuelle pour analyser les protocoles de vote dans le cas d'un intrus actif.

CHAPITRE 7

Équivalence observationnelle

Nous avons vu au chapitre 6 que l'équivalence statique permet de représenter de manière plus fine que la déduction les informations accessibles pour un intrus dans le cas passif. Dans le cas actif, c'est une notion appelée équivalence observationnelle qui permet d'exprimer des propriétés plus fines que les propriétés d'accessibilité vues aux chapitres 3, 4 et 5. Intuitivement, l'équivalence observationnelle est le pendant actif de l'équivalence statique avec l'introduction d'une bisimulation : deux processus P et Q sont en équivalence observationnelle, noté $P \sim_o Q$ si P et Q peuvent émettre sur les mêmes canaux et si tout mouvement de P peut être suivi par un mouvement de Q tels que les deux processus restent en équivalence observationnelle.

L'équivalence observationnelle permet d'exprimer des propriétés de sécurité en comparant un protocole P à sa version idéale P_{ideal} où la propriété désirée est vraie par construction. La sécurité de P s'exprime alors par la relation $P \sim_o P_{ideal}$. Des exemples sont développés dans [AG97]. L'équivalence observationnelle est aussi utilisée pour exprimer des notions comme la confidentialité d'un vote ou la résistance à la coercition pour les protocoles de vote électronique [KR05, DKR06, DKR09b]. L'équivalence observationnelle permet également d'exprimer des propriétés proches des propriétés de sécurité en cryptographie, définies à l'aide de « jeux ». Nous établirons une comparaison formelle entre ces deux notions au chapitre 9, partie 9.2.

Dans la première partie de ce chapitre, nous définissons l'équivalence observationnelle dans le cadre du pi-calcul appliqué [AF01]. Le pi-calcul appliqué est une algèbre de processus très adaptée à la modélisation des protocoles. Il permet une grande souplesse dans le choix des primitives cryptographiques, qui sont spécifiées à l'aide d'une théorie équationnelle. Nous en rappelons les principales définitions à la partie 7.1. Nous verrons à la partie 7.2 sous quelles conditions il est possible de ramener l'équivalence observationnelle à une propriété d'accessibilité, dans le cas du secret. Nous présentons à la partie 7.3 une procédure de décision pour une classe de processus, dits déterministes.

7.1 Le pi-calcul appliqué

Le pi-calcul appliqué a été défini par Martín Abadi et Cédric Fournet [AF01]. Nous en rappelons ici les principales définitions.

P, Q, R : :=	processus
$c(x).P$	réception
$\overline{c}(s).P$	émission
$\mathbf{0}$	processus de fin
$P \parallel Q$	composition parallèle
$!P$	réplication
$(\nu\alpha)P$	restriction
`if` $M = N$ `then` P `else` Q	conditionnelle

A, B, C : :=	processus étendus
P	processus
$A \| B$	composition parallèle
$(\nu\alpha)A$	restriction
$\{x \mapsto s\}$	substitution active

$s, s_1, \ldots, s_n$ sont des termes et α et c sont des noms.

FIGURE 7.1 - *Syntaxe des processus.*

7.1.1 Syntaxe

On considère une signature $\mathcal{F}$, un ensemble de variables $\mathcal{X}$ et un ensemble de noms $\mathcal{N}$. L'algèbre termes $T(\mathcal{F}, \mathcal{X}, \mathcal{X})$ est munie d'une théorie équationnelle E. De nombreux exemples ont été présentés au chapitre 6.

La syntaxe des *processus* et des *processus étendus* est définie à la figure 7.1. Par rapport au pi-calcul appliqué décrit dans [AF01], nous ne considérerons que des processus dont les canaux sont des noms (et non des variables), ce qui est suffisant pour traiter des protocoles cryptographiques. Le processus $\mathbf{0}$ est le processus qui ne fait rien, il pourra être omis. La composition parallèle permet l'exécution concurrente des processus. La restriction $(\nu\alpha)A$ correspond à la création d'un nom α frais et privé pour A. La conditionnelle `if` $M = N$ `then` P `else` Q se comporte comme P ou Q en fonction du résultat de l'évaluation du test $M = N$. Si Q est le processus nul, on pourra écrire plus simplement `if` $M = N$ `then` P. Le processus $c(x).P$ exécute P où la variable x est remplacée par le message reçu sur le canal c. Le processus $\overline{c}(s).P$ émet le message s puis se comporte comme P.

La notation $\mathsf{noms}(P)$ (resp. $\mathsf{var}(P)$) représente l'ensemble des noms (resp. variables) qui sont libres dans P. Les noms (resp. variables) liés de P, notés $\mathsf{rnoms}(P)$ (resp. $\mathsf{rvar}(P)$), peuvent être renommés. $P\{x \mapsto s\}$ est le processus P dans lequel les occurrences (libres) de x sont remplacées par s. Un *contexte d'évaluation* est une expression de la forme $C = (\nu\overline{\alpha})([\cdot] \parallel P)$ où P est un processus. Nous écrivons $C[Q]$ pour $(\nu\overline{\alpha})(Q \parallel P)$. Un contexte (resp. un processus) C est dit *clos* si $\mathsf{var}(C) = \emptyset$.

On associe à chaque processus étendu A, une structure[1] $\varphi(A)$ obtenue en remplaçant tous les processus (non étendus) par $\mathbf{0}$. Par exemple, soit $A_1 \stackrel{\mathsf{def}}{=} \nu y, k, r.\{^{\mathsf{enc}(m,k,r)}/_x, {}^{a}/_y\} \| \overline{c}(y)$ alors $\varphi(A_1) = \nu y, k, r.\{^{\mathsf{enc}(m,k,r)}/_x, {}^{a}/_y\}$.

[1] Définie page 59.

$$
\begin{array}{rcll}
A \parallel \mathbf{0} & \equiv & A & \\
A \parallel B & \equiv & B \parallel A & \\
(A \parallel B) \parallel C & \equiv & A \parallel (B \parallel C) & \\
(\nu\alpha)(\nu\beta)A & \equiv & (\nu\beta)(\nu\alpha)A & \\
(\nu\alpha)(A \parallel B) & \equiv & A \parallel (\nu\alpha)B & \text{si } \alpha \notin \mathsf{noms}(A) \cup \mathsf{var}(A) \\
(\nu x)\{x \mapsto s\} & \equiv & \mathbf{0} & \\
(\nu\alpha)\mathbf{0} & \equiv & \mathbf{0} & \\
!P & \equiv & P \parallel !P & \\
\{x \mapsto s\} \parallel A & \equiv & \{x \mapsto s\} \parallel A\{x \mapsto s\} & \\
\{x \mapsto s\} & \equiv & \{x \mapsto t\} & \text{si } s =_E t
\end{array}
$$

FIGURE 7.2 - *Équivalence structurelle.*

7.1.2 Sémantique

L'*équivalence structurelle* permet d'identifier des processus (étendus) au comportement identique. C'est la plus petite relation sur les processus étendus, notée $\equiv$, close par application de contextes et qui satisfait les relations de la figure 7.2. Par exemple, $\varphi(A_1) \equiv \nu k, r.\{^{\mathsf{enc}(m,k,r)}/_{x}\}$.

Les évolutions des processus (étendus) sont représentées par la relation $\rightarrow$, qui est la plus petite relation telle que :

$$
\begin{array}{rrcll}
\text{(Com)} & c(x).P \parallel \overline{c}(s).Q & \rightarrow & P \parallel Q & \\
\text{(Cond1)} & \texttt{if } M = N \texttt{ then } P \texttt{ else } Q & \rightarrow & P & \text{if } M =_E N \\
\text{(Cond2)} & \texttt{if } M = N \texttt{ then } P \texttt{ else } Q & \rightarrow & Q & \text{if } M \neq_E N
\end{array}
$$

$\xrightarrow{*}$ est la plus petite relation transitive sur les processus (étendus) qui contient $\equiv$ et $\rightarrow$ et qui est close par application de contextes.

Définition 7.1 *L'*équivalence observationnelle*, notée $\sim_o$, est la plus grande relation symétrique $\mathcal{S}$ sur les processus clos étendus telle que $A\mathcal{S}B$ implique :*

1. *si, pour un contexte C, un terme s et un processus A', $A \xrightarrow{*} C[\overline{c}(s) \cdot A']$ alors il existe un contexte C', un terme s' et un processus B', tels que $B \xrightarrow{*} C'[\overline{c}(s') \cdot B']$.*
2. *si $A \xrightarrow{*} A'$ alors il existe B' tel que $B \xrightarrow{*} B'$ et $A'\mathcal{S}B'$*
3. *$C[A]\mathcal{S}C[B]$ pour tout contexte d'évaluation C.*

7.1.3 Sémantique étiquetée

La quantification universelle sur tout contexte (condition 3) rend difficile les preuves d'équivalence observationnelle entre les processus. Aussi, Martín Abadi et Cédric Fournet [AF01] ont proposé une sémantique alternative et équivalente, appelée *sémantique étiquetée*. La relation de réduction *étiquetée*, notée $\xrightarrow{\alpha}$, étend la relation $\rightarrow$ par les règles de la figure 7.3.

$$c(x).P \xrightarrow{c(M)} P\{^{M}/_{x}\} \quad \text{(IN)} \qquad \overline{c}u.P \xrightarrow{\overline{c}u} P \quad \text{(OUT-ATOM)}$$

$$\frac{A \xrightarrow{\overline{c}u} A'}{\nu u.A \xrightarrow{\nu u.\overline{c}u} A'}\, u \neq c \quad \text{(OPEN-ATOM)} \qquad \frac{A \xrightarrow{\alpha} A'}{\nu u.A \xrightarrow{\alpha} \nu u.A'}\, u \notin \alpha \quad \text{(SCOPE)}$$

$$\frac{A \xrightarrow{\alpha} A'}{A\|B \xrightarrow{\alpha} A'\|B}\,(*) \quad (PAR) \qquad \frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'} \quad \text{(STRUCT)}$$

où c est un nom, u est un nom ou une variable, et la condition (*) de la règle (PAR) est $\mathsf{rvar}(\alpha) \cap \mathsf{var}(B) = \mathsf{rnoms}(\alpha) \cap \mathsf{noms}(B) = \emptyset$.

FIGURE 7.3 - *Sémantique étiquetée.*

Deux processus P et Q sont en bisimulation étiquetée si les séquences de messages déjà échangées par P et Q sont statiquement équivalentes et si tout mouvement de P peut être suivi d'un mouvement de Q tels que les deux processus restent en bisimulation étiquetée.

Définition 7.2 *La* bisimulation étiquetée *($\approx_l$) est la plus grande relation $\mathcal{R}$ sur les processus clos étendus telle que $A\,\mathcal{R}\,B$ implique :*

1. *$\varphi(A) \approx \varphi(B)$;*
2. *si $A \to A'$ alors $B \to^* B'$ et $A'\,\mathcal{R}\,B'$, pour un certain B' ;*
3. *si $A \xrightarrow{\alpha} A'$ avec $\mathsf{var}(\alpha) \subseteq \mathsf{dom}(\varphi(A))$ et $\mathsf{rnoms}(\alpha) \cap \mathsf{noms}(B) = \emptyset$ alors $B \to^* \xrightarrow{\alpha} \to^* B'$ et $A'\,\mathcal{R}\,B'$, pour un certain B'.*

La notion de bisimulation étiquetée est plus facile à manipuler. C'est la notion qui sera utilisée dans le reste de ce chapitre. Martín Abadi et Cédric Fournet ont montré qu'elle coïncide avec l'équivalence observationnelle.

Théorème 7.1 ([AF01]) $\approx_l = \sim_o$

La réduction étiquetée permet de définir les traces d'un processus étendu. Soit $\mathcal{A}$ l'alphabet (infini) des actions étiquetant la réduction étiquetée $\xrightarrow{\alpha}$, définie à la figure 7.3. Pour tout mot $a_1 \cdots a_n \in \mathcal{A}^*$ la relation $\overset{a_1 \cdots a_n}{\Rightarrow}$ est définie par $\to^* \xrightarrow{a_1} \to^* \cdots \to^* \xrightarrow{a_n} \to^*$. Le mot $a_1 \cdots a_n$ correspond à la séquence d'actions visibles d'un attaquant. Une *trace* d'un processus étendu est une séquence d'actions visibles ainsi que la séquence des messages échangés durant l'exécution.

$$\mathsf{trace}(A) = \{(s, \varphi(B)) \mid A \overset{s}{\Rightarrow} B \text{ pour un certain } B\}.$$

7.1.4 Exemple

Reprenons l'exemple du protocole Wide-Mouthed Frog. Une session du rôle A joué par l'agent a pour l'agent b peut être modélisée par le processus $\nu\{r, k_{ab}\}.A(a, b, c_a)$ où $A(a, b, c_a)$ est défini par

$$A(a, b, c_a) \overset{\mathsf{def}}{=} \overline{c_a}(\mathsf{pair}(a, \mathsf{enc}(\mathsf{pair}(b, k_{ab}), k_{as}, r))).$$

De même, une session du rôle S répondant à a pour b peut être modélisée par le processus $\nu r.S(a,b,c_s)$ où $S(a,b,c_s)$ est défini par

$$S(a,b,c_s) \stackrel{\mathsf{def}}{=} c_s(x).\texttt{if } \mathsf{fst}(x) = a \wedge \mathsf{fst}(\mathsf{dec}(\mathsf{snd}(x), k_{as})) = b \\ \texttt{then } \overline{c_s}(\mathsf{enc}(\mathsf{pair}(b, \mathsf{snd}(\mathsf{dec}(\mathsf{snd}(x), k_{as}))), k_{bs}, r)) \\ \texttt{else } \mathbf{0}$$

Pour modéliser un nombre borné d'exécution du rôle A et du rôle S, il suffit de répliquer les processus.

$$P_1 \stackrel{\mathsf{def}}{=} !(\nu\{r, k_{ab}\}.A(a,b,c_a)) \parallel !(\nu r.S(a,b,c_s))$$

Pour une modélisation plus complète, il faudrait bien sûr également considérer des instances du rôle A avec d'autres agents : $A(b,a)$, $A(c,b)$, $S(a,b)$, $S(c,b)$, *etc.*

7.2 Lien entre secret fort et secret faible

L'équivalence observationnelle permet d'exprimer une notion plus forte du secret. Ainsi, nous dirons que le nom s est *fortement secret* dans P si, pour tous termes clos M, M' tels que $\mathsf{rnoms}(P) \cap (\mathsf{noms}(M) \cup \mathsf{noms}(M')) = \emptyset$,

$$P[^M/_s] \sim_o P[^{M'}/_s]$$

où $P[^M/_s]$ représente l'instantiation du nom s par M dans P. Intuitivement, un intrus ne doit pas être capable de distinguer deux exécutions du protocole avec des valeurs différentes pour la donnée s, même si ces valeurs sont publiques. C'est une notion plus forte que la définition habituelle du secret puisqu'il est possible que l'intrus distinguer deux exécutions avec des valeurs différentes pour le secret, sans pour autant connaître la valeur du secret.

Cette définition a été proposée par Martín Abadi et Andrew Gordon [AG97]. Elle est plus proche de la notion d'indistinguabilité pour définir la confidentialité d'une donnée. Nous comparerons ces deux notions au chapitre 9.2. Si la notion de secret fort offre plus de garanties de sécurité, elle est également plus difficile à prouver. À notre connaissance, le seul outil capable de traiter le secret fort est l'outil ProVerif. L'outil ProVerif [Bla05] n'est pas directement capable de prouver l'équivalence observationnelle mais cherche à prouver une notion plus précise, qui implique l'équivalence observationnelle [BAF05, BAF08].

La notion plus habituelle du secret, utilisée dans les chapitres 3, 4 et 5 sera appelée *secret syntaxique* : le nom s est *syntaxiquement secret* dans P si pour toute trace (t, φ) de $\mathsf{trace}(P)$, s n'est pas déductible à partir de φ : $\varphi \not\vdash_E s$.

On montre facilement que le secret fort implique le secret syntaxique. La réciproque n'est bien sûr pas vraie en général. Le but de cette partie est d'identifier dans quel cadre le secret syntaxique suffit à assurer le secret fort. Cela permet en particulier l'utilisation des outils développés pour le secret syntaxique tout en obtenant de meilleures garanties (secret fort).

7.2.1 Algèbre

Dans le reste de cette partie, la signature considérée est $\mathcal{F} = \{\mathsf{enc}, \mathsf{dec}, \mathsf{enca}, \mathsf{deca}, \mathsf{pub}, \mathsf{priv}, \mathsf{pair}, \mathsf{fst}, \mathsf{snd}, \mathsf{sign}, \mathsf{checksign}, \mathsf{ok}, \mathsf{getsign}\}$, munie de la théorie équationnelle E induite

par les équations :

$$\begin{aligned}
\mathsf{fst}(\mathsf{pair}(x,y)) &= x\\
\mathsf{snd}(\mathsf{pair}(x,y)) &= y\\
\mathsf{dec}(\mathsf{enc}(z_1,z_2,z_3),z_2) &= z_1\\
\mathsf{deca}(\mathsf{enca}(z_1,\mathsf{pub}(z_2),z_3),\mathsf{priv}(z_2)) &= z_1\\
\mathsf{checksign}(z_1,\mathsf{sign}(z_1,\mathsf{priv}(z_2)),\mathsf{pub}(z_2)) &= \mathsf{ok}\\
\mathsf{getsign}(\mathsf{sign}(z_1,z_2)) &= z_1
\end{aligned}$$

Remarquons en particulier que les symboles de chiffrement enc et enca sont cette fois ternaires, le troisième argument représentant l'aléa utilisé lors du chiffrement. La théorie équationnelle E sera sous-entendue dans la suite de cette partie.

7.2.2 Cas passif

Nous commençons par étudier dans quels cas le secret syntaxique implique le secret fort dans le cas passif :

$$\varphi \not\vdash s \quad \stackrel{?}{\Rightarrow} \quad \varphi[M/s] \approx \varphi[M'/s] \quad \forall M, M'$$

L'implication n'est bien sûr pas vraie en général. Nous en avons identifié plusieurs raisons *[CRZ06, CRZ07]*.

Chiffrement probabiliste. La structure $\psi_1 = \nu s,k,r.\{^{\mathsf{enc}(s,k,r)}/_x, {}^{\mathsf{enc}(n,k,r)}/_y\}$ ne préserve pas le secret fort de s car les deux chiffrés utilisent le même aléa, ce qui permet à un intrus de comparer les chiffrés : $\psi_1[n/s] \not\approx \psi_1[n'/s]$ car $(x=y)\,\psi_1[n/s]$ et $(x\neq y)\,\psi_1[n'/s]$.

Secret en position de clef. La structure $\psi_2 = \nu s,n.\{^{\mathsf{enc}(\langle n,n'\rangle,s,r)}/_x\}$ ne préserve pas le secret fort de s car s apparaît en position de clef, ce qui permet à un intrus d'essayer de déchiffrer : $\psi_2[k/s] \not\approx \psi_2[k'/s]$ car $(\pi_2(\mathsf{dec}(x,k)) = n')\,\psi_2[k/s]$ et $(\pi_2(\mathsf{dec}(x,k)) \neq n')\,\psi_2[k'/s]$.

Présence de destructeurs. La structure $\psi_3 = \nu s.\{^{\pi_1(s)}/_x\}$ ne préserve pas le secret fort de s car elle révèle une information partielle sur s : $(x=k)$ est vrai $\psi_3[\langle k,k'\rangle/s]$ mais pas pour $\psi_3[k/s]$.

Règle getsign. L'équation $\mathsf{getsign}(\mathsf{sign}(z_1,z_2)) = z_1$ peut paraître arbitraire car les schémas de signature ne permettent pas nécessairement de calculer le message signé à partir de la signature. Cette équation est pourtant cruciale pour notre résultat comme l'illustre l'exemple suivant. La structure $\psi_4 = \nu s.\{^{\mathsf{sign}(s,\mathsf{priv}(a))}/_x, {}^{\mathsf{pub}(a)}/_y\}$ ne préserve pas le secret fort de s. En effet, l'égalité $(\mathsf{checksign}(n,x,y) = \mathsf{ok})$ est vraie pour $\psi_4[n/s]$ mais pas pour $\psi_4[n'/s]$.

Dans les trois premiers cas, les structures proposées préservent le secret syntaxique de s : $\psi_i \not\vdash s$, pour $1 \leq i \leq 3$. Dans le quatrième cas, nous aurions également $\psi_4 \not\vdash s$ sans l'équation $\mathsf{getsign}(\mathsf{sign}(z_1,z_2)) = z_1$.

Ces exemples nous ont conduits à introduire la notion de bonne formation. Une structure φ est *bien formée* vis-à-vis d'un nom s si :

1. le chiffrement est probabiliste, *i.e.* un aléa différent est utilisé dans chaque terme chiffré ;
2. le nom s n'apparaît ni en position clef, ni en position d'aléa ;
3. φ ne contient pas les symboles dec, deca, fst, snd, checksign, getsign, dits *destructeurs*.

La condition de bonne formation suffit à assurer que le secret syntaxique implique le secret fort dans le cas passif.

Théorème 7.2 (*[CRZ06, CRZ07]*) *Soit φ une structure bien formée vis-à-vis de s.*

$$\nu s\varphi \not\vdash s \quad \textit{si et seulement si} \quad \varphi[^{M}/_{s}] \approx \varphi[^{M'}/_{s}]$$

pour tous termes clos M, M' tels que $\mathsf{rnoms}(\varphi) \cap (\mathsf{noms}(M) \cup \mathsf{noms}(M')) = \emptyset$.

Pour préparer l'étude du cas actif, nous avons étendu ce résultat aux structures contenant des destructeurs, à condition que les destructeurs ne soient pas appliqués au dessus du nom s, ni au-dessus d'un constructeur *[CRZ06, CRZ07]*.

7.2.3 Cas actif

Dans le cas actif, il faut également tenir compte des tests effectués dans les conditionnelles. Considérons par exemple le processus

$$P_1 = \nu s, k, r.\big(\overline{c}(\mathsf{enc}(s,k,r)) \mid c(z).[\mathsf{dec}(z,k) = a].\overline{c}(\mathsf{ok})\big)$$

où a est un nom libre. Alors s est syntaxiquement secret dans P_1 mais n'est pas fortement secret. En effet, $P_1 \to P_1' \stackrel{\mathsf{def}}{=} \nu s, k, r.(\{^{\mathsf{enc}(s,k,r)}/_{z}\} \mid [s = a].\overline{c}(\mathsf{ok}))$ et le processus $P_1'[^{a}/_{s}]$ peut encore être réduit, alors que ce n'est pas le cas de $P_1'[^{b}/_{s}]$.

Nous avons donc étendu la notion de bonne formation aux processus P de telle manière que :

1. les termes de P sont bien formés ;
2. les tests des conditionnelles ne portent pas sur s.

Pour assurer ce dernier point quelque soit le comportement du protocole, nous calculons (automatiquement) une sur-approximation des messages envoyés et nous vérifions qu'aucun des tests ne peut porter sur le nom s.

La condition de bonne formation sur les processus permet d'assurer que le secret syntaxique implique le secret fort dans le cas actif.

Théorème 7.3 (*[CRZ06, CRZ07]*) *Soit P un processus bien formé vis-à-vis de s. Alors $\varphi \not\vdash s$ pour toute trace (t, φ) de $\mathsf{trace}(\nu s P)$ si et seulement si $P[^{M}/_{s}] \approx_l P[^{M'}/_{s}]$, pour tous termes clos M, M' tels que* $\mathsf{rnoms}(P) \cap (\mathsf{noms}(M) \cup \mathsf{noms}(M')) = \emptyset$.

Ce résultat nous a en particulier permis de prouver le secret fort des clefs échangées dans le protocole de Yahalom [BAN89, CJ97], dans celui de Needham-Schroeder à clef symétrique [NS78] ainsi que pour une version modifiée du protocole Wide-Mouthed Frog.

7.3 Procédure de décision

Nous avons proposé une procédure de décision, basée sur la procédure proposée par Mathieu Baudet [Bau05, Bau07], pour une classe particulière de processus, appelés *simples*. Pour cela, nous avons montré trois résultats intermédiaires.

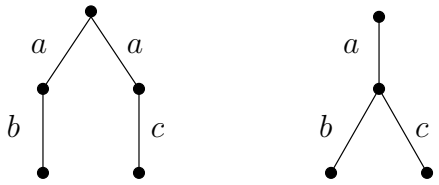


FIGURE 7.4 - *L'égalité des traces ne coïncide pas avec la bisimulation en général.*

1. Pour les processus déterministes, l'équivalence observationnelle coïncide avec l'équivalence de traces. Ce résultat s'inspire d'un résultat similaire dans le cadre du pi-calcul [Eng85].
2. La plupart des protocoles cryptographiques peuvent être modélisés par des processus *simples*, qui sont déterministes.
3. L'équivalence de traces est décidable pour les processus sans réplication. Ce résultat s'appuie sur la procédure proposée par Mathieu Baudet [Bau05, Bau07].

7.3.1 Équivalence de traces

Nous dirons que deux processus sont en équivalence de traces si leurs ensembles de traces coïncident, à équivalence statique près.

Définition 7.3 (Équivalence de traces) *Soient A et B deux processus étendus clos. Nous écrivons $A \sqsubseteq_t B$ si pour chaque $(s, \varphi) \in \mathsf{trace}(A)$ tel que $\mathsf{rnoms}(s) \cap \mathsf{noms}(B) = \emptyset$, il existe $(s, \varphi') \in \mathsf{trace}(B)$ tel que $\varphi \approx \varphi'$.*

Les processus A et B sont en équivalence de traces, noté $A \approx_t B$, si $A \sqsubseteq_t B$ et $B \sqsubseteq_t A$.

L'équivalence observationnelle implique l'équivalence de traces mais il est bien connu que la réciproque est fausse en générale. Ceci est dû au fait que deux processus peuvent avoir le même ensemble de traces sans être bisimilaires, comme l'illustre la figure 7.4.

J. Engelfriet a montré dans le cadre du pi-calcul [Eng85] que l'équivalence de traces coïncide avec l'équivalence observationnelle pour les processus *déterministes*. Nous avons étendu ce résultat au pi-calcul appliqué.

Un processus est déterministe si pour une trace donnée, il n'y a qu'une exécution possible.

Définition 7.4 (Déterminisme) *Soit $\cong$ une relation d'équivalence sur les processus étendus clos. Un processus étendu clos A est* $\cong$-déterministe *si pour tous B, B' tels que $A \stackrel{s}{\Rightarrow} B$, $A \stackrel{s}{\Rightarrow} B'$ et $\varphi(B) \approx \varphi(B')$ alors $B \cong B'$.*

Plusieurs choix sont possibles pour la relation d'équivalence. Nous avons montré *[CD09a]* que le $\sim_o$-déterminisme et le $\approx_l$-déterminisme coïncident. Aussi, dans la suite de cette partie, nous dirons tout simplement qu'un processus est déterministe dès qu'il est $\sim_o$-déterministe ou $\approx_l$-déterministe. Nous avons montré que l'équivalence de traces coïncide avec l'équivalence observationnelle pour les processus déterministes.

Théorème 7.4 (*[CD09a]*) *Soient A et B deux processus étendus clos déterministes.*

$$A \approx_t B \text{ si et seulement si } A \sim_o B.$$

L'équivalence de traces n'est pas nécessairement plus facile à montrer que l'équivalence observationnelle. Cependant, dans le cadre des protocoles cryptographiques, la plupart des résultats existants portent sur des propriétés de traces. Nous illustrons ainsi l'intérêt de ce théorème à la partie 7.3.3, en nous ramenant à une procédure de décision existante.

7.3.2 Processus simples

La plupart des protocoles sont déterministes : lorsqu'un agent reçoit un message, il sait exactement comment le traiter. Nous avons donc identifié *[CD09a]* la classe des processus *simples* qui suffit à modéliser par exemple les protocoles de la bibliothèque Clark&Jacob [CJ97].

Définition. Les processus simples sont construits à partir de processus basiques. Un processus est *basique, émettant sur le canal* c, s'il est de la forme :
- $\mathsf{if}\ M = N\ \mathsf{then}\ \overline{c}(s_1).B_1\ \mathsf{else}\ \overline{c}(s_2).B_2$
- ou $c(x) \cdot B$

où B, B_1, B_2 sont des processus basiques (ou nuls), émettant sur le canal c. Intuitivement, un processus basique permet de décrire un rôle d'un protocole.

Les processus *simples* sont obtenus en composant et restreignant les processus basiques.

$$\nu\tilde{n}.\ (\quad \begin{array}{ll} \nu\tilde{n}_1.(B_1 \mid \sigma_1) & \mid \quad !(\nu c'_1, \tilde{m}_1.\overline{p_1}(c'_1).B'_1) \\ \vdots & \vdots \\ \nu\tilde{n}_k.(B_k \mid \sigma_k) & \mid \quad !(\nu c'_n, \tilde{m}_n.\overline{p_n}(c'_n).B'_n) \end{array} \quad)$$

où chacun des processus simples $B_1, \ldots, B_b, B'_1, \ldots, B'_n$ émet sur un canal distinct des autres canaux d'émission et c'_i est le canal associé au processus B'_i. Les processus simples permettent d'exprimer l'exécution de plusieurs rôles en parallèle, un nombre borné ou non borné de fois. Leur particularité est que chaque session d'un rôle du protocole se voit attribuer un nom de canal particulier.

Exemple. Ainsi les processus $A(a, b, c_a)$ et $S(a, b, c_s)$ décrits page 74 sont des processus basiques. Le processus P_1 n'est pas un processus simple car le même canal est utilisé pour des sessions différentes. Mais il est possible de représenter un nombre non borné d'exécutions des rôles A et S par le processus simple ci-dessous.

$$P_2 \stackrel{\mathsf{def}}{=} !(\nu\{r, k_{ab}, c_a\}.\overline{p_A}(c_a).A(a, b, c_a)) \parallel !(\nu\{r, c_s\}.\overline{p_S}(c_s).S(a, b, c_s))$$

Théorème 7.5 (*[CD09a]*) *Tout processus simple est déterministe.*

Le point clef pour s'assurer du déterminisme est que chaque instance d'un rôle (une session) émet et reçoit un nom de canal particulier, rendu public pour qu'un attaquant puisse intercepter les messages et en envoyer de nouveaux. Il est intéressant de remarquer que les codages habituels de protocoles utilisent un seul canal public pour tous les processus. Ainsi, considérons un rôle qui attend un input puis émet ok ainsi qu'un rôle qui attend un input puis émet ko. La composition de ces deux rôles est en générale modélisée par le processus

$$P_2 \stackrel{\mathsf{def}}{=} c(x).\overline{c}(\mathsf{ok}) \parallel c(x).\overline{c}(\mathsf{ko})$$

Cette modélisation n'est pas déterministe puisque $P_2 \overset{c(M)}{\Rightarrow} P_2' \overset{\mathsf{def}}{=} \overline{c}(\mathsf{ok}) \parallel c(x).\overline{c}(\mathsf{ko})$ et $P_2 \overset{c(M)}{\Rightarrow} P_2'' \overset{\mathsf{def}}{=} c(x).\overline{c}(\mathsf{ok}) \parallel \overline{c}(\mathsf{ko})$ et $P_2' \not\sim_o P_2''$. Nous pensons cependant que la modélisation classique, consistant à n'utiliser qu'un seul canal public pour toutes les communications, affaiblit les capacités d'un attaquant En effet, l'utilisation d'un canal unique cache à l'attaquant l'identité exacte de l'émetteur du message et empêche l'attaquant de contrôler à qui les messages sont envoyés. Ainsi, supposons que la propriété de sécurité souhaitée est qu'il n'existe pas d'attaquant capable de forcer *toutes* les exécutions à n'émettre que la constante ok. Alors la modélisation du protocole à l'aide du processus P_2 permettra de démontrer qu'il n'y a pas d'attaque possible puisque certaines traces conduiront à l'émission de la constante ko. Pourtant, sur un réseau comme Internet, un attaquant peut parfaitement choisir la destination de ses messages. Ainsi, une modélisation utilisant des canaux (publics) distincts pour chaque rôle et chaque session nous semble mieux adaptée. En revenant à notre exemple, sa modélisation devient ainsi le processus ci-dessous,

$$c_1(x).\overline{c_1}(\mathsf{ok}) \parallel c_2(x).\overline{c_2}(\mathsf{ko})$$

qui est un processus simple.

7.3.3 Décidabilité de l'équivalence observationnelle

Pour obtenir une procédure de décision pour l'équivalence observationnelle, nous nous sommes restreints à des processus sans réplication puisque les propriétés de secret et d'authentification sont indécidables pour un nombre non borné de sessions. Nous avons également considéré des processus sans branche `else`, c'est-à-dire que tels que tout sous processus de la forme `if` $M = N$ `then` P `else` Q vérifie que $Q = \mathbf{0}$.

Ces deux restrictions sont suffisantes pour obtenir la décidabilité de l'équivalence de traces pour les processus simples et donc de l'équivalence observationnelle, pour des théories équationnelles sous-termes convergentes[2].

Théorème 7.6 (*[CD09a]*) *Soit E une théorie sous-terme convergente. Soient A et B deux processus simples sans réplication ni branche* `else`*. L'équivalence observationnelle de A et B est un problème co-NP-complet.*

Le fragment des processus simples sans réplication ni branche `else` avec théories sous-terme convergentes, permet par exemple de modéliser tous les protocoles de la bibliothèque Clark& Jacob, pour un nombre borné de sessions.

L'ensemble des traces d'un processus simple, même sans réplication, est infini. Il est cependant possible d'en donner une représentation symbolique finie, telle que deux processus simples sont en équivalence de traces si et seulement si ils sont en équivalence symbolique de traces. Ces travaux s'inspirent fortement du calcul symbolique proposé par S. Delaune, S. Kremer et M. Ryan [DKR07, DKR09a]. Nous avons ensuite montré que, pour les processus simples sans réplication ni branche `else`, l'équivalence symbolique de traces se ramène à tester l'équivalence d'un nombre fini de systèmes de contraintes. Nous pouvons ensuite conclure à l'aide du résultat de Mathieu Baudet [Bau05, Bau07] qui assure que l'équivalence de systèmes de contraintes est co-NP-complet pour les théories équationnelles sous-termes convergentes.

[2] Définies page 61

7.4 Perspectives

Il existe encore très peu de procédures de décision pour l'équivalence observationnelle. H. Hüttel a proposé une procédure de décision pour un fragment sans réplication du spi-calcul [Hut02] (avec des primitives cryptographiques fixes). L. Durante, R. Sisto et A. Valenzano ont également proposé un fragment décidable pour la "testing equivalence", plus faible que l'équivalence observationnelle [DSV03]. D'autres approches consistent à mettre au point des relations d'équivalence plus fortes, qui impliquent l'équivalence observationnelle, comme des bisimulations symboliques [BBN04, BN05, JPVB08, DKR07, DKR09a] ou l'équivalence de bi-processus [BAF05, BAF08]. La seule procédure implémentée est l'équivalence de bi-processus proposée par Bruno Blanchet et traitée par l'outil ProVerif. Cependant, l'équivalence de bi-processus échoue pour montrer l'équivalence observationnelle dans le cadre des protocoles de vote. Ceci est dû au fait que les processus ne peuvent pas être (bi)simulés en suivant les mêmes branches d'exécutions [DKR09a].

Notre résultat de décidabilité pour les processus simples sans réplication ni branche `else` permet de traiter exactement l'équivalence observationnelle. Cependant, pour traiter complètement les protocoles de vote, il faudrait étendre ce résultat dans plusieurs directions. Tout d'abord, il faudrait traiter des processus avec une primitive de synchronisation, qui permet de modéliser les différentes phases d'un vote ou d'un jeu cryptographique par exemple. Cette extension semble relativement aisée à réaliser, en modifiant la sémantique symbolique associée.

D'autre part, il faudrait traiter des théories équationnelles plus larges pour prendre en compte les primitives cryptographiques utilisées dans les protocoles de vote, comme les théories E_{DVP} ou E_{Trap} présentées au chapitre 6.5. Il faudrait pour cela adapter le résultat de Mathieu Baudet [Bau05, Bau07] à des théories équationnelles plus riches. Nous avons vu au chapitre 6 comment décider la déduction et l'équivalence statique pour E_{DVP} ou E_{Trap}. Il faudrait étendre ce résultat dans le cadre d'un intrus actif (ce qui est probablement non trivial). Une autre extension consiste à traiter des processus avec branches `else`. Il faudrait à nouveau adapter le résultat de Mathieu Baudet [Bau05, Bau07] à des systèmes de contraintes avec inégalités.

Pour implémenter la procédure de décision proposée dans ce chapitre, il faudrait en premier lieu implémenter la procédure de Mathieu Baudet. Il s'agirait donc d'étendre l'outil YAPA (partie 6.3.2) au cas actif. L'implémentation d'une procédure pour l'équivalence observationnelle devrait alors en découler facilement.

Troisième partie

Correction des modèles symboliques par rapport aux modèles calculatoires

CHAPITRE 8

Correction des propriétés de traces

Les chapitres précédents de ce mémoire sont consacrés à l'étude des protocoles cryptographiques dans le cadre de modèles symboliques, où les primitives cryptographiques sont représentées par des symboles fonctionnels, éventuellement munis d'une théorie équationnelle. Nous avons vu en introduction (partie 1.3.3) que ces modèles sont très différents de ceux utilisés en cryptographie où les messages sont représentés par des suites de bits et l'adversaire est n'importe quelle machine de Turing polynomiale.

Le travail précurseur de Martín Abadi et Philipp Rogaway [AR00] a montré que les modèles symboliques permettent de prouver l'indistinguabilité cryptographique dans le cas passif. Dans le cas actif, Michael Backes *et al.* ont proposé une bibliothèque symbolique permettant d'abstraire les opérations cryptographiques [SBB+06, BP05b]. Leur bibliothèque permet de traiter de nombreuses primitives (signatures et chiffrement asymétrique [BPW03], chiffrement symétrique [BP04], protocoles à divulgation nulle [BU08], etc.). Peter Laud a également proposé une procédure de décision symbolique pour des preuves cryptographiques de protocoles à chiffrement symétrique [Lau04]. John Mitchell *et al.* ont proposé une logique symbolique (PCL) [DDM+05, DDMW06, RDDM07] qui permet d'obtenir des garanties dans les modèles cryptographiques. Ces travaux ne permettent cependant pas de conclure quant aux garanties offertes par les modèles symboliques habituels.

Dans ce chapitre, nous décrivons plusieurs résultats montrant que les modèles symboliques utilisés jusqu'ici permettent *directement* d'obtenir des garanties cryptographiques. Cette approche fait suite au travail précurseur de Daniele Micciancio et Bogdan Warinschi [MW04b].

8.1 Modèle

Nous présentons le modèle commun aux résultats de ce chapitre.

8.1.1 Syntaxe

Pour faire le lien avec les modèles calculatoires où les messages sont des suites de bits, il est nécessaire d'enrichir un peu l'algèbre de termes habituellement utilisée. Nous considérons un ensemble (infini) A de noms d'agents, des ensembles infinis de noms Nonce_{ag}, Nonce_{adv}, Rand_{ag}, et Rand_{adv} (représentant respectivement les nonces et les aléas des agents honnêtes et de l'adversaire), ainsi qu'un ensemble infini $\mathsf{Garbage}$ représentant les suites de bits ne correspondant à aucun message valide. Soit $\mathsf{Nonce} = \mathsf{Nonce}_{ag} \cup \mathsf{Nonce}_{adv}$ et $\mathsf{Rand} = \mathsf{Rand}_{ag} \cup \mathsf{Rand}_{adv}$.

$\dfrac{}{\phi \vdash m}\ m \in \phi$	$\dfrac{\phi \vdash b}{\phi \vdash \mathsf{pub}(b), \phi \vdash \mathsf{vk}(b)}\ b \in \mathsf{A} \cup X.a$	Connaissance initiale
$\dfrac{\phi \vdash m_1 \quad \phi \vdash m_2}{\phi \vdash \mathsf{pair}(m_1, m_2)}$	$\dfrac{\phi \vdash \mathsf{pair}(m_1, m_2)}{\phi \vdash m_i}\ i \in \{1, 2\}$	Concaténation et projection
$\dfrac{\phi \vdash \mathsf{pub}(b), \phi \vdash m}{\phi \vdash \{m\}^r_{\mathsf{pub}(b)}}\ r \in \mathsf{Rand}_{adv}$	$\dfrac{\phi \vdash \{m\}^r_{\mathsf{pub}(b)} \quad \phi \vdash \mathsf{priv}(b)}{\phi \vdash m}$	Chiffrement et déchiffrement
$\dfrac{\phi \vdash m}{\phi \vdash h(m)}$		Hachage
$\dfrac{\phi \vdash \mathsf{sk}(b), \phi \vdash m}{\phi \vdash \mathsf{sign}(m, \mathsf{sk}(b), r)}\ r \in \mathsf{Rand}_{adv}$	$\dfrac{\phi \vdash \mathsf{sign}(m, \mathsf{sk}(b), r)}{\phi \vdash m}$	Signature

FIGURE 8.1 - *Règles de déduction.*

Nous considérons également un ensemble fini de variables X, qui contient en particulier des variables d'agents A_i, $i \in \{1, \ldots, k\}$, des variables de nonces $X^j_{A_i}$ ainsi que des variables d'aléa $L^j_{A_i}$ $j \in \mathbb{N}$, qui représentent respectivement les nonces et l'aléa de chiffrement généré par l'agent A_i.

L'ensemble des messages est défini par la grammaire

$$\begin{aligned} \mathsf{M} \quad ::= \quad & \mathsf{A} \mid \mathsf{Nonce} \mid \mathsf{pub(A)} \mid \mathsf{priv(A)} \mid \mathsf{vk(A)} \mid X \mid \mathsf{pair(M, M)} \\ & \mid \{\mathsf{M}\}^{\mathsf{Rand}}_{\mathsf{pub(A)}} \mid \mathsf{sign(M, sk(A), Rand)} \mid h(\mathsf{M}) \mid \mathsf{Garbage} \end{aligned}$$

En particulier, le message $\{m\}^r_{\mathsf{pub}(a)}$ représente le message m chiffré par la clef publique $\mathsf{pub}(a)$ de l'agent a avec l'aléa r. Le symbole de chiffrement est ternaire pour représenter le chiffrement probabiliste. Si l'aléa $r \in \mathsf{Rand}_{adv}$, alors le message a été construit par l'adversaire et donc m est connu de l'adversaire. Le système de déduction associé est défini à la figure 8.1. Toutes les règles sont habituelles sauf peut-être la règle

$$\frac{\phi \vdash \mathsf{sign}(m, \mathsf{sk}(b), r)}{\phi \vdash m}$$

qui permet à un attaquant de calculer un message à partir de sa signature. Suivant l'implémentation du schéma de chiffrement, la signature d'un message révèle des informations partielles sur le message. Cette règle renforce donc le pouvoir d'un attaquant symbolique.

Les *rôles* d'un protocole sont représentés par une séquence de la forme $(l_1, r_1) \cdots (l_k, r_k)$ où r_i et l_i sont des termes (avec des variables). Ils permettent d'exprimer des séquences de réception et d'émission où r_i représente le message attendu et l_i représente le message envoyé. Un protocole à k participants (ou plutôt à k rôles) est une fonction $\Pi : \{1, \ldots, k\} \to \mathsf{Roles}$ où Roles représente l'ensemble des rôles.

Exemple. Considérons l'exemple du protocole Wide-Mouthed Frog.

– Le rôle A est représenté par la séquence d'une seule règle

$$(\mathsf{init}, \mathsf{pair}(A_1, \{\mathsf{pair}(A_2, X^1_{A_1})\}^{L^1_{A_1}}_{K(A_1)}))$$

– Le rôle S est représenté par la séquence d'une seule règle

$$(\mathsf{pair}(A_1, \{\mathsf{pair}(A_2, X^1_{A_1})\}^{L^1_{A_1}}_{K(A_1)}), \mathsf{pair}(\{\mathsf{pair}(A_1, X^1_{A_1})\}^{L^1_S}_{K(A_2)}))$$

– Le rôle B est représenté par la séquence d'une seule règle

$$(\mathsf{pair}(\{\mathsf{pair}(A_1, X^1_{A_1})\}^{L^1_S}_{K(A_2)})), \mathsf{stop})$$

où init et stop sont des constantes spéciales marquant le début ou la fin d'un rôle. Cet exemple est fourni à titre d'illustration même si l'algèbre de termes considérée ici ne permet pas d'exprimer le chiffrement symétrique.

8.1.2 Exécution

Étant donné un protocole Π, nous définissons (informellement) l'ensemble de ses traces symboliques, noté $\mathsf{Exec}^s(\Pi)$ ainsi que l'ensemble de ses traces concrètes.

Dans les deux cas, l'adversaire (symbolique ou calculatoire) peut interagir avec le protocole par l'intermédiaire de trois types de requêtes.

corrupt$(a_1, \ldots, a_l)$ En début d'interaction, l'adversaire peut corrompre un ensemble d'agents $a_1, \ldots, a_l$. En retour, il reçoit leurs clefs (symboliques, c'est-à-dire des termes, ou concrètes, c'est-à-dire des suites de bits, suivant le modèle d'exécution).

new$(i, a_1, \ldots, a_k)$ L'adversaire peut initier une nouvelle session pour le rôle i où l'agent a_i cherche à communiquer avec les agents $a_1, \ldots, a_k$. On ajoute alors un nouvel état local pour l'agent a_i, où chaque variable de nonce X_{a_i} et d'aléa L_{a_i} est initialisée avec des valeurs fraîches (des noms frais dans le modèle symbolique et des valeurs générées aléatoirement dans le modèle calculatoire)

send$((a_i, s), m)$ L'adversaire peut envoyer un nouveau message m à l'agent a_i pour la session s.

– Pour le modèle symbolique, il faut que le terme m soit déductible à partir des messages précédemment échangés, pour la notion de déduction définie à la figure 8.1.

– Pour le modèle calculatoire, il faut que le message m soit calculé par l'adversaire, qui est une machine de Turing probabiliste polynomiale.

Si le message m correspond au message l_j attendu, alors le message r_j correspondant (c'est-à-dire, dans le modèle symbolique, le terme $r_j\theta$ avec θ telle que $l_j\theta = m$) est donné à l'adversaire et l'état local de l'agent est mis à jour en fonction du message reçu. Si m ne correspond pas au message attendu, l'état local de l'agent est inchangé.

L'exécution concrète est paramétrée par le *paramètre de sécurité* η qui fixe la longueur des clefs et des nonces. L'adversaire concret est n'importe quelle machine de Turing $\mathcal{A}$ probabiliste polynomiale en le paramètre de sécurité. Si on suppose fixé l'aléa $R_{\mathcal{A}}$ de l'adversaire ainsi que l'aléa R_Π du protocole, la trace concrète obtenue par l'interaction de $\mathcal{A}$ avec Π est uniquement déterminée et est notée $\mathsf{Exec}_{\Pi(R_\Pi),\mathcal{A}(R_{\mathcal{A}})}(\eta)$.

8.1.3 Interprétation des termes

À chaque symbole fonctionnel (pair, $\{\}$, sign, h) correspond une fonction cryptographique. Lorsqu'on cherche à relier les modèles symboliques et concrets, il faut tout d'abord mettre en relation les noms de type nonces et aléas avec des suites de bits. Étant donné un terme t et une *interprétation* des nonces et des aléas de t, c'est-à-dire une fonction qui envoie les nonces et les aléas de t sur des suites de bits, l'interprétation de t, notée $[\![t]\!]$ est fixée : il suffit de calculer l'interprétation du terme t en partant de ses feuilles et en appliquant pas à pas les fonctions cryptographiques associées à chaque symbole fonctionnel. On dit qu'une trace concrète t_c est l'*image* d'une trace symbolique t_s s'il existe une interprétation des nonces et des aléas de t_s telle que $t_c = [\![t_s]\!]$.

À l'inverse, toute suite de bits sera interprétée par un terme (quitte à utiliser des noms de Garbage). Pour faciliter la lecture d'une suite de bits, nous supposerons que chaque suite de bits commence par une *étiquette* qui indique son type (nom, clef, concaténation, chiffrement, signature). Toute suite de bits commençant par une étiquette invalide sera invalide. Ces étiquettes peuvent bien sûr être modifiées par l'attaquant. Cependant, à chaque réception de messages, les agents vérifient que les suites de bits sont étiquetées par le type attendu.

8.1.4 Quelques notions de cryptographie

La modélisation des messages à l'aide de termes suppose que les primitives cryptographiques soient très robustes. Ainsi, le chiffrement ne doit pas être malléable (un adversaire ne peut changer le contenu d'un message chiffré sans avoir la clef) et un intrus ne doit pas obtenir d'information sur le contenu d'un message chiffré.

Sécurité des primitives. Dans la suite de ce chapitre ainsi que dans le chapitre suivant, nous ferons appel à principalement deux notions de sécurité pour le chiffrement : IND-CPA et IND-CCA. Intuitivement, un schéma de chiffrement est IND-CPA si un attaquant ne peut pas relier un message chiffré $\{m_i\}_k$, $i \in \{0,1\}$ au message en clair m_i, même si m_0 et m_1 ont été choisis par l'attaquant et si l'attaquant à accès à un oracle de chiffrement. Un schéma de chiffrement est IND-CCA si un attaquant ne peut pas relier $\{m_i\}_k$, $i \in \{0,1\}$ à m_i, même si m_0 et m_1 ont été choisis par l'attaquant et si l'attaquant à accès à un oracle de chiffrement et de déchiffrement. Une définition précise est donnée dans l'article [BBM00].

Pour les signatures, nous supposerons souvent que le schéma de signature est « existentially unforgeable » [GMR88], c'est-à-dire qu'un attaquant ne peut pas créer une signature valide sans avoir la clef de signature, même en ayant accès à un oracle de signature.

Les fonctions de hachage sont difficiles à abstraire dans le mode symbolique [BP05a]. Nous supposerons ici qu'elles sont implémentées dans le *modèle de l'oracle aléatoire* [BR93]. Le modèle de l'oracle aléatoire consiste à supposer l'existence d'un oracle qui associe à chaque message un nombre aléatoire : étant donnée une requête m, l'oracle génère un nombre aléatoire n, renvoie n et mémorise l'association (n, m). Si la requête m lui est à nouveau envoyée plus tard, il renverra le nombre n. Cette implémentation n'est bien sûr pas réaliste mais c'est une hypothèse classique pour obtenir des preuves cryptographiques en présence de fonctions de hachage.

Définition de la confidentialité calculatoire. Dans les modèles cryptographiques, la confidentialité d'une donnée comme un nonce X_{A_i} pour un protocole Π est définie à l'aide d'une

expérience $\mathbf{Exp}^{b,X_{A_i}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(\eta)$ paramétrée par un bit $b \in \{0,1\}$. Deux nonces n_0 et n_1 sont générés pour X_{A_i}. Dans l'expérience $\mathbf{Exp}^{b,X_{A_i}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(\eta)$ l'adversaire interagit avec le protocole Π où X_{A_i} est instancié par n_b. Lorsque l'adversaire décide de stopper l'interaction avec le protocole, on lui donne n_0 et n_1. Puis l'adversaire doit deviner la valeur de b, il produit la valeur b'. Le protocole Π *préserve calculatoirement le secret* de X_{A_i} s'il n'est pas capable de distinguer les deux expériences, c'est-à-dire si

$$\Pr\left[\mathbf{Exp}^{1,X_{A_i}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(\eta) = 1\right] - \Pr\left[\mathbf{Exp}^{0,X_{A_i}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(\eta) = 1\right]$$

est une fonction *négligeable* de η.

Une fonction $f : \mathbb{R} \to \mathbb{R}$ est *négligeable* si elle est ultimement plus petite que l'inverse de tout polynôme.

$$\forall \text{ polynôme } P,\ \exists \eta_0,\ \forall \eta \geqslant \eta_0, \quad |f(\eta)| \leqslant \frac{1}{|P(\eta)|}$$

8.2 Chiffrement asymétrique et signatures

Dans cette partie, nous considérons uniquement des protocoles avec concaténation, chiffrement asymétrique et signatures, c'est-à-dire des protocoles formés sur l'algèbre de termes définie à la section 8.1.1, privée du symbole h pour les fonctions de hachage.

Nous avons montré que toute trace concrète est l'image d'une trace symbolique.

Théorème 8.1 (*[CW05]*) *Soit* Π *un protocole. Si le schéma de chiffrement est IND-CCA et que la signature est « existentially unforgeable » alors pour toute machine de Turing polynomiale probabiliste* $\mathcal{A}$*, toute trace concrète est l'image d'une trace symbolique, sauf avec probabilité négligeable.*

$$\Pr\left[\exists t^s \in \mathsf{Exec}^s(\Pi) \mid [\![t^s]\!] = \mathsf{Exec}^c_{\Pi(R_\Pi),\mathcal{A}(R_\mathcal{A})}(\eta)\right] \geq 1 - \nu_\mathcal{A}(\eta)$$

où $\nu_\mathcal{A}$ *une fonction négligeable.*

Ce résultat permet de transférer immédiatement toutes les propriétés de traces : si un protocole vérifie symboliquement une propriété de traces, comme l'authentification, définie par exemple à l'aide de la logique $\mathcal{L}$ définie au chapitre 3.2.1 alors cette propriété est également vraie dans le monde cryptographique.

Ce résultat n'est cependant pas complètement satisfaisant dans le cas d'une propriété de secret. En effet, au niveau symbolique, le secret d'une donnée est défini comme l'incapacité pour un attaquant de produire cette donnée. Le théorème 8.1 nous permet donc de déduire immédiatement que si un protocole préserve le secret symbolique, alors aucune machine de Turing polynomiale probabiliste ne sera capable d'émettre le secret, sauf avec une probabilité négligeable. Cette notion de secret est beaucoup plus faible que la notion définie au paragraphe 8.1.4. Nous avons montré que le secret symbolique suffit cependant à assurer le secret cryptographique, dans le cadre du chiffrement asymétrique et des signatures.

Théorème 8.2 (*[CW05]*) *Soit* Π *un protocole. Si le schéma de chiffrement est IND-CCA et que la signature est « existentially unforgeable » alors si* Π *préserve symboliquement le secret d'un nonce* X_{A_i}*,* Π *préserve calculatoirement le secret de nonce* X_{A_i}*.*

8.3 Extension aux fonctions de hachage

Nous avons étendu les résultats précédents au cas où les protocoles comportent des fonctions de hachage. Dans ce cadre, il est clair que le secret symbolique ne peut plus suffire à assurer le secret calculatoire. Considérons par exemple le cas d'un protocole qui révèle le hachage d'un nonce : $h(n)$. Alors ce protocole préserve (symboliquement) le secret de n alors qu'il n'en préserve pas le secret calculatoire. En effet, si l'adversaire observe $h(n_b)$, n_0 et n_1, il peut aisément déduire la valeur de b en calculant $h(n_0)$ et $h(n_1)$ et en comparant au message $h(n_b)$ observé.

Nous avons proposé une nouvelle définition de secret symbolique, en calculant la partie accessible d'un message, appelée *patron*. Pour simplifier les démonstrations, nous nous sommes restreints aux protocoles avec concaténations, chiffrement asymétrique et fonctions de hachage, mais sans signatures.

Définition 8.1 (Patrons) *Soient $S = \{M_1, M_2, \ldots, M_k\}$ un ensemble de termes clos et T un terme, nous définissons $\mathsf{Pat}_T(S) = \{\mathsf{Pat}_T^S(M_1), \mathsf{Pat}_T^S(M_2), \ldots, \mathsf{Pat}_T^S(M_k)\}$, où $\mathsf{Pat}_T^S(M)$ est défini récursivement par :*

$$
\begin{array}{rcl}
\mathsf{Pat}_T^S(a) & = & \begin{cases} a & si\ S, T \vdash a \\ \square & sinon \end{cases} \\
\mathsf{Pat}_T^S(\langle M_1, M_2 \rangle) & = & \langle \mathsf{Pat}_T^S(M_1), \mathsf{Pat}_T^S(M_2) \rangle \\
\mathsf{Pat}_T^S(\{M\}^r_{\mathsf{pub}(a)}) & = & \begin{cases} \{\mathsf{Pat}_T^S(M)\}^r_{\mathsf{pub}(a)} & si\ S, T \vdash \mathsf{priv}(a)\ ou\ si\ r \in \mathsf{Rand}_{adv} \\ \square & sinon \end{cases} \\
\mathsf{Pat}_T^S(h(M)) & = & \begin{cases} h(\mathsf{Pat}_T^S(M)) & si\ S, T \vdash M \\ \square & sinon \end{cases}
\end{array}
$$

Intuitivement, $\mathsf{Pat}_T(S)$ représente ce qui est accessible à partir de S, en connaissant T. Si tr est une trace symbolique, on note $\mathsf{Pat}_T(\mathsf{tr})$ l'ensemble $\mathsf{Pat}_T(S)$ où S est l'ensemble des messages échangés dans la trace tr. Nous retrouvons la définition habituelle du secret en déclarant qu'un nonce X_{A_i} est secret dans un protocole Π si, pour toute trace $\mathsf{tr} \in \mathsf{Exec}^s(\Pi)$, pour toute instance n de X_{A_i}, le nonce n n'apparaît pas dans $\mathsf{Pat}_\emptyset(\mathsf{tr})$. Pour assurer le secret calculatoire, nous demandons que le nonce n reste invisible, *même si l'intrus connaît n*. Plus précisément, nous dirons que le nonce X_{A_i} est *invisible* dans le protocole Π si, pour toute trace $\mathsf{tr} \in \mathsf{Exec}^s(\Pi)$, pour toute instance n de X_{A_i} dans la trace tr, le nonce n n'apparaît pas dans $\mathsf{Pat}_n(\mathsf{tr})$.

Nous retrouvons ainsi que n apparaît dans $\mathsf{Pat}_n(h(n)) = h(n)$: le message $h(n)$ ne rend pas n invisible. Par contre, n n'apparaît pas dans $\mathsf{Pat}_n(h(\mathsf{pair}(n, n'))) = \square$: le nonce n' permet de cacher n puisque l'adversaire ne peut pas reconstruire $h(\mathsf{pair}(n, n'))$.

Théorème 8.3 (*[CKKW06]*) *Soit Π un protocole. Si le schéma de chiffrement est IND-CCA et que la fonction de hachage est implémentée dans le modèle de l'oracle aléatoire, alors si le nonce X_{A_i} est invisible dans le protocole Π alors Π préserve calculatoirement le secret de nonce X_{A_i}.*

D'autre part, nous avons montré que notre nouvelle notion de secret (invisibilité) est décidable pour un nombre borné de sessions.

Théorème 8.4 (*[CKKW06]*) *Le problème suivant est NP-complet :*
Donnée *C un système de contraintes*[1] *et un nom n,*
Question *existe-t-il une solution θ de C telle que n apparaît dans* $\mathsf{Pat}_n(C\theta)$ *?*

8.4 Conclusion et Perspectives

Les résultats de ce chapitre montrent qu'il est possible d'utiliser les procédures de décision et les outils développés pour l'analyse des protocoles cryptographiques, pour obtenir automatiquement des résultats dans les modèles cryptographiques. Cela permet ainsi de faire l'économie des preuves longues et difficiles, habituellement faites à la main dans les modèles calculatoires [War03, War05].

Pour rapport aux modèles habituellement utilisés pour modéliser symboliquement les protocoles, l'algèbre de termes est plus riche. En particulier, le symbole de chiffrement est ternaire au lieu de binaire et nous considérons un symbole explicite pour la signature alors que la signature d'un agent A est souvent codée à l'aide d'un chiffrement avec la clef privée de A. Nous avons cependant montré la correction des codages utilisés pour simplifier l'analyse automatique effectuée par des outils comme Avispa : si un protocole est sûr en oubliant le troisième argument du chiffrement et en codant la signature à l'aide du chiffrement, alors il est sûr dans l'algèbre de termes présentée dans ce chapitre, pour une large classe de propriétés *[CHW07]*. Nous avons ainsi intégré à l'outil Avispa[2] un module permettant de détecter (automatiquement) que la preuve (symbolique) de sécurité effectuée par l'outil permet de déduire directement une preuve cryptographique.

Un premier développement naturel de ce chapitre est d'unifier et d'élargir les primitives cryptographiques utilisées. Il serait intéressant d'obtenir au minimum un résultat de correction pour des protocoles avec chiffrement asymétrique, concaténation, signatures et fonctions de hachage. Cela demande en particulier d'étendre la notion de *patrons* mise au point pour définir l'invisibilité dans le cadre des signatures. Il serait également naturel de considérer de nouvelles primitives cryptographiques, comme le chiffrement symétrique. Nous verrons cependant au prochain chapitre qu'adapter le théorème 8.1 au chiffrement symétrique pose des problèmes particuliers.

Le travail de ce chapitre a été repris par Michael Backes et Dominique Unruh pour traiter le cas des protocoles à divulgation nulle dans un cadre symbolique [BU08]. D'autre part, Roberto Segala et Andrea Turrini ont étudié la traduction de notre résultat de correction (partie 8.2) dans le cadre des automates probabilistes avec entrées/sorties (PIOA) [CCK+08, Tur09].

De nombreux autres développements, communs avec le chapitre suivant, seront détaillés au chapitre 10.

[1] Défini page 29.
[2] `http://avispa-project.org/`

CHAPITRE 9

Correction des propriétés d'équivalence

Dans les modèles cryptographiques, la plupart des notions de sécurité s'énonce comme l'équivalence (ou l'indistinguabilité) de différents « jeux ». Ainsi, nous avons vu au chapitre 8.1.4 que la confidentialité s'exprime comme l'incapacité, pour un adversaire, de distinguer une exécution d'un protocole où le secret est instancié par n_0, d'une exécution où le secret est instancié par n_1. De la même manière, l'indistinguabilité de jeux est à la base de la définition de l'anonymat dans les signatures de groupe [AW04] ou de la « cécité » des signatures [JLO97]. Toujours dans le cadre des modèles calculatoires, la notion très utilisée de *simulation forte* (« strong simulatability » [BPW07, KT08]) s'exprime de la manière suivante : un protocole P simule une fonctionnalité idéale F s'il existe un simulateur S tel que P et $S\|F$ ne peuvent être distingués par un environnement. Différentes variantes de la notion de composition universelle [Can01, CR03, CH06] s'exprime également à l'aide d'indistinguabilité.

Dans le cadre symbolique, nous avons défini l'équivalence statique (chapitre 6) ainsi que l'équivalence observationnelle (chapitre 7) qui permettent également d'exprimer l'indistinguabilité *symbolique* dans le cas actif et passif.

L'objet de ce chapitre est d'étudier sous quelles hypothèses l'équivalence statique et l'indistinguabilité de messages coïncident dans le cas d'un attaquant passif (partie 9.1) et sous quelles hypothèses l'équivalence observationnelle et l'indistinguabilité de protocoles coïncident dans le cas d'un attaquant actif (partie 9.2).

9.1 Correction de l'équivalence statique

Dans les modèles symboliques, les messages sont représentés à l'aide d'une algèbre de termes, munie d'une théorie équationnelle qui reflète les propriétés des primitives cryptographiques. Ainsi, les équations

$$\begin{array}{rclcrcl} x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & \qquad & x \oplus y & = & y \oplus x \\ x \oplus x & = & 0 & & x \oplus 0 & = & x \end{array}$$

reflètent certaines propriétés de l'opérateur « ou exclusif ». À chaque fois qu'une théorie équationnelle est fixée, plusieurs questions sont naturelles : pouvons-nous être sûrs d'avoir considéré toutes les propriétés pertinentes ? À l'inverse, avons-nous ajouté trop d'équations ?

Ces questions nous ont amenés à définir six notions *[BCK05]* qui mesurent l'adéquation de la théorie équationnelle aux primitives cryptographiques considérées.

Une théorie équationnelle E est dite
- $=_E$-*correcte* si et seulement si $T_1 =_E T_2$ implique que $\Pr[e_1, e_2 \leftarrow [\![T_1, T_2]\!]; e_1 \neq e_2]$ est négligeable ;
- $=_E$-*complète* si et seulement si $T_1 \neq_E T_2$ implique que $\Pr[e_1, e_2 \leftarrow [\![T_1, T_2]\!]; e_1 = e_2]$ est négligeable ;
- $\approx_E$-*correcte* si et seulement si pour toutes structures φ_1 et φ_2, $\varphi_1 \approx_E \varphi_2$ implique que les distributions correspondantes $[\![\varphi_1]\!]$ et $[\![\varphi_2]\!]$ sont indistinguables, ce qui est noté $[\![\varphi_1]\!] \approx [\![\varphi_2]\!]$;
- $\approx_E$-*complète* si et seulement si $\varphi_1 \not\approx_E \varphi_2$ implique $[\![\varphi_1]\!] \not\approx [\![\varphi_2]\!]$;
- $\not\vdash_E$-*correcte* si et seulement si $\varphi \not\vdash_E T$ implique que pour tout adversaire polynomial $\mathcal{A}$, $\Pr[\phi, e \leftarrow [\![\varphi, T]\!]; e \leftarrow \mathcal{A}(\phi)]$ est négligeable ;
- $\not\vdash_E$-*complète* si et seulement si $\varphi \vdash_E T$ implique l'existence d'un adversaire polynomial $\mathcal{A}$ tel que $1 - \Pr[\phi, e \leftarrow [\![\varphi, T]\!]; e \leftarrow \mathcal{A}(\phi)]$ est négligeable.

Nous avons montré que ces notions ne sont pas indépendantes, même pour des théories équationnelles générales *[BCK05]*. En particulier, si la théorie équationnelle contient au moins un symbole libre h et que h est implémenté par une fonction de hachage résistante aux collisions [Sch96b], alors la correction de l'équivalence statique $\approx_E$ implique les cinq autres notions. Nous avons également proposé des critères pour prouver la correction de l'équivalence statique.

À l'aide de ces critères, nous avons montré la correction de l'équivalence statique pour deux théories (et deux implémentations) particulières : la théorie du « ou exclusif » et une théorie pour le chiffrement symétrique.

Proposition 9.1 (*[BCK05]*) *L'implémentation usuelle de la théorie équationnelle du XOR est $=_{E_\oplus}$-, $\approx_{E_\oplus}$- et $\not\vdash_{E_\oplus}$-correcte. Elle est également $=_{E_\oplus}$-, $\approx_{E_\oplus}$- et $\not\vdash_{E_\oplus}$-complète.*

Pour le chiffrement symétrique, nous avons considéré une implémentation déterministe. Comme la longueur d'un chiffré dépend de la longueur du message à chiffrer, nous avons considéré une structure de liste, ainsi qu'un opérateur de chiffrement enc_n pour chaque longueur $n \in \mathbb{N}$ possible de liste. La théorie équationnelle correspondante E_{sym} est définie par les équations suivantes :

$$\begin{aligned} \mathsf{dec}_n(\mathsf{enc}_n(x,y),y) &= x & \qquad \mathsf{cons}_n(\mathsf{head}_n(x),\mathsf{tail}_n(x)) &= x \\ \mathsf{enc}_n(\mathsf{dec}_n(x,y),y) &= x & \qquad \mathsf{enc}_0(\mathsf{nil},x) &= \mathsf{nil} \\ \mathsf{head}_n(\mathsf{cons}_n(x,y)) &= x & \qquad \mathsf{dec}_0(\mathsf{nil},x) &= \mathsf{nil} \\ \mathsf{tail}_n(\mathsf{cons}_n(x,y)) &= y \end{aligned}$$

Proposition 9.2 (*[BCK05]*) *Soient φ_1 et φ_2 deux structures avec clefs atomiques, sans cycle de clef et sans symbole* head *ni* tail. *Si l'implémentation du schéma de chiffrement est SRP [PP04], alors $\varphi_1 \approx_{E_{\mathsf{sym}}} \varphi_2$ implique $[\![\varphi_1]\!] \approx [\![\varphi_2]\!]$.*

La notion de correction de l'équivalence statique a été généralisée par Steve Kremer et Laurent Mazaré [KM07]. Ils définissent ainsi la « correction de l'équivalence statique adaptative » où l'adversaire peut choisir au fur et à mesure la séquence de messages qu'il souhaite voir. Cette définition est *a priori* plus appropriée pour servir de base au cas actif.

9.2 Correction de l'équivalence observationnelle

La partie précédente était consacrée au cas d'un intrus *passif*, qui observe des séquences de messages. Dans le cas d'un intrus actif, un candidat naturel pour représenter symboliquement

l'indistinguabilité cryptographique est l'équivalence observationnelle. Nous avons montré que pour un fragment très proche des processus simples définis au chapitre 7.3.2, l'équivalence observationnelle implique l'indistinguabilité cryptographique. Nous nous sommes restreints au cas du chiffrement symétrique (et de la concaténation). Pour éviter de considérer des termes contenant des destructeurs explicites, nous avons enrichi le pi-calcul appliqué [AF01] avec de nouveaux prédicats.

9.2.1 Pi-calcul appliqué

Extension du pi-calcul appliqué. Soit $\mathcal{P}$ est un ensemble de symboles de prédicats. La syntaxe considérée dans cette partie est la même que celle du pi-calcul appliqué (définie au chapitre 7.1, figure 7.1) sauf que nous autorisons désormais des conditionnelles de la forme `if` ϕ `then` P `else` Q où ϕ est une *condition*, définie par la grammaire ci dessous.

ϕ, ψ `::=`	conditions
$p(s_1, \ldots, s_n)$	application d'un prédicat $p \in \mathcal{P}$
$\phi \wedge \psi$	conjonction

Si l'ensemble de prédicats $\mathcal{P}$ est réduit au seul prédicat d'égalité $=$ modulo la théorie équationnelle, on retrouve alors exactement le pi-calcul appliqué défini dans [AF01].

L'ajout de prédicats nous conduit à étendre la définition de l'équivalence statique : deux structures sont statiquement équivalentes si elles satisfont les mêmes prédicats.

Définition 9.1 (Équivalence statique avec prédicats) *Soit ϕ une structure, p un prédicat et $s_1, \ldots, s_k$ des termes. Nous dirons que ϕ satisfait l'expression $p(s_1, \ldots, s_k)$, noté $\phi \models p(s_1, \ldots, s_k)$, si il existe des noms $\tilde{n}$ tels que $\phi = \nu\tilde{n}.\sigma$, $\mathsf{noms}(s_i) \cap \overline{n} = \emptyset$ pour tout $1 \leq i \leq k$ et $p(s_1, \ldots, s_k)\sigma$ est vrai. Nous dirons que deux structures $\phi_1 = \nu\overline{n}.\sigma_1$ et $\phi_2 = \nu\overline{n'}.\sigma_2$ sont* statiquement équivalentes, *noté $\phi_1 \approx \phi_2$ si $\mathsf{dom}(\phi_1) = \mathsf{dom}(\phi_2)$, et*

$$\forall s_1, \ldots, s_k \in T(\mathcal{N}, \mathcal{X}), \forall p \in \mathcal{P}, \quad \phi_1 \models p(s_1, \ldots, s_k) \Leftrightarrow \phi_2 \models p(s_1, \ldots, s_k).$$

Nous retrouvons à nouveau la définition habituelle de l'équivalence statique en considérant $\mathcal{P}$ réduit au seul prédicat d'égalité $=$ modulo la théorie équationnelle.

Signature et Prédicats. Dans la suite de ce mémoire, nous considérerons en particulier la signature $\mathcal{F}_0 = \{\mathsf{pair}, \mathsf{fst}, \mathsf{snd}, \mathsf{enc}, \mathsf{dec}\} \cup C_0$ où C_0 est un ensemble de constantes contenant en particulier une constante 0^l de longueur l pour tout $l \in \mathbb{N}$. Nous supposons en effet l'existence d'une fonction *longueur* qui est un homomorphisme des termes vers les entiers naturels. Le symbole de chiffrement est ternaire et $\mathsf{enc}(m, k, r)$ représente le message m chiffré avec la clef k et l'aléa r. La signature $\mathcal{F}_0$ est munie de la théorie équationnelle E_0 habituelle pour le chiffrement symétrique et la concaténation.

$$\begin{aligned} \mathsf{fst}(\mathsf{pair}(x, y)) &= x \\ \mathsf{snd}(\mathsf{pair}(x, y)) &= y \\ \mathsf{dec}(\mathsf{enc}(z_1, z_2, z_3), z_2) &= z_1 \end{aligned}$$

Les équations peuvent être orientées de gauche vers la droite, formant ainsi un système de réécriture convergent. Chaque terme s a une unique forme normale notée $s\downarrow$.

Nous considérons également l'ensemble de prédicats $\mathcal{P}_0 = \{M, EQ, \mathsf{P}_{\mathsf{samekey}}, EL\}$ définis de la manière suivante.

- M est unaire et est vrai sur le terme clos s si et seulement si $s\downarrow$ ne contient pas de symboles de projection (fst ou snd) ou de déchiffrement (dec).
- EQ est binaire et est vrai sur (s,t) si et seulement si $M(s)$ et $M(t)$ sont vrais et si $s\downarrow = t\downarrow$: il s'agit de l'interprétation de l'égalité modulo la théorie équationnelle E_0.
- $\mathsf{P}_{\mathsf{samekey}}$ est binaire et est vrai sur les chiffrés utilisant la même clef de chiffrement : $\mathcal{M} \models \mathsf{P}_{\mathsf{samekey}}(s,t)$ si et seulement s'il existe k,u,v,r,r' tels que

$$EQ(s,\mathsf{enc}(u,k,r)) \text{ et } EQ(t,\mathsf{enc}(v,k,r')).$$

- EL est binaire et est vrai sur (s,t) si et seulement si $M(s)$ et $M(t)$ sont vrais et si s et t ont la même longueur.

Processus simples. La classe des processus traitée dans cette partie est très proche de la famille des processus simples définie à la partie 7.3.2. Les deux principales différences sont les suivantes :

- À la différence des processus simples de la partie 7.3.2, toutes les communications s'effectuent sur les canaux c_{in} (pour la réception) et c_{out} (pour l'émission). Mais chaque message envoyé est de la forme $\mathsf{pair}(l,m)$ où l identifie le rôle et la session où le message m est envoyé. Chaque réception d'un message commence par la vérification que l'identifiant l reçu correspond bien à l'identifiant de la session.
- Les conditionnelles sont de la forme $\mathsf{if}\ \phi\ \mathsf{then}\ \overline{c}(s).B\ \mathsf{else}\ \overline{c}(\bot).\mathbf{0}$: les processus ne doivent pas comporter de branches else.

Ainsi, un processus simple correspondant à un nombre non borné d'exécutions des rôles A et S du protocole Wide-Mouthed Frog est

$$P_3 \stackrel{\mathsf{def}}{=} \nu(k_{as},k_{bs})\ \ (\ \ !((\nu k_{ab},,r,l)\overline{c_{\mathsf{out}}}(l).A'(a,b))\|\ !((\nu r,l)\overline{c_{\mathsf{out}}}(l).S'(a,b,l))\ \ )$$

où $A'(a,b)$ et $S'(a,b,l)$ sont définis ci-dessous.

$$A'(a,b) = \overline{c_{\mathsf{out}}}(\mathsf{pair}(l,\mathsf{pair}(a,\mathsf{enc}(\mathsf{pair}(b,k_{ab}),k_{as},r)))) \cdot \mathbf{0}$$

$$\begin{aligned} S'(a,b,l) = c_{\mathsf{in}}(x).\ &\mathtt{if}\ \ EQ\ (\pi_1(x),l)\\ &\quad \mathtt{then\ if}\ \ \pi_1(\pi_2(x)) = a\ \wedge\ \pi_1(\mathsf{dec}(\pi_2(\pi_2(x)),k_{as})) = b\\ &\qquad \mathtt{then}\ \overline{c_{\mathsf{out}}}(\mathsf{pair}(l,\mathsf{enc}(\mathsf{pair}(a,\pi_2(\mathsf{dec}(\pi_2(\pi_2(x)),k_{as}))),k_{bs},r))) \cdot \mathbf{0}\\ &\qquad \mathtt{else}\ \ \overline{c_{\mathsf{out}}}(\bot) \cdot \mathbf{0}\\ &\quad \mathtt{else}\ \overline{c_{\mathsf{out}}}(\bot) \cdot \mathbf{0} \end{aligned}$$

D'autre part, nous supposons que pour tout sous-terme de la forme $\mathsf{enc}(t,k,v)$ apparaissant dans un processus simple, v est un nom restreint qui n'apparaît dans aucun autre sous-terme distinct. Cette restriction permet cependant plusieurs occurrences du sous-terme $\mathsf{enc}(t,k,v)$, à la différence de [AR00, AR07].

9.2.2 L'équivalence observationnelle implique l'indistinguabilité

L'interprétation des termes en tant que suites de bits a été définie au chapitre précédent. Les processus simples peuvent facilement être interprétées par des machines de Turing communicantes proches des machines de Turing interactives définies par Ran Canetti [Can01]. Chaque processus est interprété par une machine de Turing polynomiale avec un ruban spécial pour les nombres aléatoires (*ruban aléatoire*). Le paramètre de sécurité η est initialement

écrit sur le ruban d'entrée de chaque machine. Nous supposons également l'existence d'un environnement qui distribue les éventuelles clefs partagées. Toutes les communications sont contrôlées et synchronisées par l'attaquant. L'interprétation d'un processus P (simple) est notée $[\![P]\!]$.

Deux machines de Turing communicantes, représentées respectivement par les processus P et Q sont *indistinguables*, noté $[\![P]\!] \approx [\![Q]\!]$, si un attaquant ne peut distinguer une interaction avec $[\![P]\!]$ d'une interaction avec $[\![Q]\!]$. Plus formellement $[\![P]\!]$ et $[\![Q]\!]$ sont indistinguables si la probabilité

$$\Pr\left[\, [\![P]\!](\eta) \parallel \mathcal{A}(\eta) = 1 \,\right] - \Pr\left[\, [\![Q]\!](\eta) \parallel \mathcal{A}(\eta) = 1 \,\right]$$

est négligeable.

Le but de cette partie est d'étudier sous quelles conditions l'équivalence observationnelle est une abstraction correcte de l'indistinguabilité de processus.

9.2.2.1 Hypothèses

Nous décrivons ci-dessous les principales hypothèses de notre résultat, avant de l'énoncer.

Chiffrement IND-CPA et IND-CTXT Nous supposons que le schéma de chiffrement est IND-CPA[1] (ou plus précisément « type 3-secure » pour la définition proposée dans [AR07]) et IND-CTXT, défini dans [BN00]. La notion IND-CTXT assure qu'il est impossible à un attaquant de forger un chiffrement valide sans avoir la clef de chiffrement. Cela implique en particulier que le déchiffrement d'un chiffré échoue (presque) toujours si la clef de déchiffrement ne correspond pas à la clef de chiffrement [MW04a].

Génération honnête des clefs Nous supposerons que l'attaquant peut obtenir de nouvelles clefs symétriques uniquement par l'intermédiaire d'un serveur, qui assure que les clefs (mêmes malhonnêtes) sont engendrées à l'aide du schéma de génération de clefs. À la réception d'une nouvelle clef, les agents en vérifient la validité (à l'aide de la signature du serveur par exemple). Cette hypothèse n'est pas très réaliste car elle suppose l'existence d'un serveur de confiance pour les clefs symétriques, ce qui n'est pas le cas en pratique. Cependant, cette hypothèse est cruciale pour la validité de notre résultat et nous proposons plusieurs types de contre-exemples à la partie 9.2.3.

Hiérarchie sur les clefs Nous supposons l'existence d'un ordre sur les clefs tel qu'une clef ne chiffre jamais une clef plus grande qu'elle. Nous avons vu au chapitre 3 que cette propriété est décidable pour un nombre borné de sessions (proposition 3.2).

Parsing Comme au chapitre 8, nous supposons que les suites de bits sont étiquetées de manière à faciliter leur (unique) interprétation par des termes.

9.2.2.2 Résultat

Nous avons montré que, sous ces hypothèses, l'équivalence observationnelle est une abstraction correcte de la notion d'indistinguabilité.

[1] Défini page 88.

Théorème 9.1 (*[CLC08a]*) *Soient P_1 et P_2 deux processus simples. Alors $P_1 \sim_o P_2$ implique $[\![P_1]\!] \approx [\![P_2]\!]$.*

Ce résultat est en particulier valable pour des processus simples avec réplication. Dans le mode symbolique, le nombre de sessions peut être arbitraire. Cependant, dans le modèle cryptographique, l'adversaire étant une machine de Turing polynomiale, il ne pourra initier qu'un nombre (au plus) polynomial de sessions.

Ainsi, notre résultat permet par exemple de prouver symboliquement la propriété (cryptographique) d'anonymat dans les signatures de groupe [AW04]. Cette propriété s'énonce (intuitivement) de la manière suivante. Soit $P(x, y)$ un protocole de signature, qui signe le message x avec la clef de signature de l'agent y. Un attaquant ne doit pas pouvoir distinguer le cas où l'agent a_1 signe du cas où l'agent a_2 signe. Cette propriété s'exprime par l'équivalence ci-dessous :

$$P_1 = c(y).P(\pi_1(y), \pi_1(\pi_2(y))) \sim_o P_2 = c(y).P(\pi_1(y), \pi_2(\pi_2(y))).$$

Intuitivement, l'adversaire enverra le message $\mathsf{pair}(m, \mathsf{pair}(a_1, i_2))$ où m est le message à signer et a_1, a_2 sont deux identités. P_1 signe m avec la clef de a_1 alors que P_2 signe m avec la clef de a_2. Ces deux protocoles doivent être indistinguables cryptographiquement. D'après le théorème 9.1, il suffit de montrer que P_1 et P_2 sont en équivalence observationnelle.

L'équivalence observationnelle peut être prouvée à l'aide de sur-approximations [BBN04, BN05, JPVB08, DKR07, DKR09a], en particulier par l'outil ProVerif [Bla05]. Nous avons également proposé une procédure de décision au chapitre 7 (théorème 7.6) pour les processus simples sans réplication. Cependant, aucun de ces résultats ne s'appliquent exactement à la classe des processus simples considérée dans ce chapitre. Nous discutons les adaptations possibles à la partie 9.3.

9.2.2.3 Étapes de la preuve.

Soient deux processus (simples) P_1 et P_2 tels que $P_1 \sim_o P_2$. On pourrait croire qu'il suffit d'une part de montrer qu'il est possible de relever les traces concrètes en des traces symboliques (comme pour le théorème 8.1) et d'autre part, de montrer la correction de l'équivalence statique. Ainsi, pour toute trace concrète t_1^c de $[\![P_1]\!]$, il existerait une trace symbolique t_1 de P_1. Comme $P_1 \sim_o P_2$, on pourrait déduire qu'il existe une trace t_2 de P_2 telle que $t_1 \approx t_2$. En invoquant la correction de l'équivalence statique, on pourrait alors conclure $[\![t_1]\!] \approx [\![t_2]\!]$ avec $t_1^c \in [\![t_1]\!]$. Il n'est cependant pas possible d'aller plus loin car chacune des traces concrètes de $[\![P_1]\!]$ peut correspondre à une trace symbolique différente. Ainsi la proportion de traces de $[\![P_1]\!]$ dans $[\![t_1]\!]$ peut être négligeable pour chaque trace t_1 de P_1 et il est impossible de conclure.

Nous avons introduit la notion d'*arbre de calcul* T_P associé à un processus P. L'arbre de calcul T_P associé à P est l'arbre des traces symboliques de P. Nous avons de plus défini une notion d'équivalence symbolique $\sim$ et calculatoire $\approx$ pour les arbres de calcul. La preuve du théorème 9.1 utilise les arbres de calcul comme étape intermédiaire pour montrer les trois implications suivantes :

$$P \sim_o Q \Rightarrow T_P \sim T_Q \Rightarrow T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$$

$P \sim_o Q \Rightarrow T_P \sim T_Q$: Ce résultat est vrai pour toute algèbre de termes, quelque soit la théorie équationnelle. Cela revient à montrer que l'équivalence observationnelle implique

la bisimulation étiquetée[2] dans un cadre un peu différent de [AF01] mais il s'agit de l'implication la plus facile.

$T_P \sim T_Q \Rightarrow T_P \approx T_Q$: La preuve utilise la notion de « tree soundness », un nouveau concept qui généralise la correction de l'équivalence statique à des adversaires qui peuvent adapter les messages choisis. Cette notion est proche (mais plus forte) de la notion « adaptive static equivalence » définie par [KM07].

$T_P \approx T_Q \Rightarrow [\![P]\!] \approx [\![Q]\!]$ Ce résultat utilise principalement le relèvement des traces concrètes en des traces symboliques. Nous avons donc montré l'équivalent du théorème 8.1 dans le cadre du chiffrement symétrique.

9.2.3 Discussion des hypothèses

Comme énoncé à la partie 9.2.2.1, la correction de l'équivalence observationnelle vis à vis de l'indistinguabilité de processus suppose quatre principales hypothèses : chiffrement IND-CPA et IND-CTXT, étiquetage des suites de bits, pas de cycles de clefs et génération honnête des clefs. Les deux premières hypothèses sont facilement réalisables, il suffit de choisir une implémentation adaptée à l'aide d'un schéma de chiffrement suffisamment fort comme XCBC [GD02] ou OCB [RBB03]. La troisième hypothèse (pas de cycles de clefs) revient à restreindre la classe des protocoles considérés et il est possible de tester à l'avance qu'un protocole ne peut pas produire de cycles de clefs (proposition 3.2).

La dernière hypothèse (génération honnête des clefs) n'est par contre pas satisfaisante, en particulier dans le contexte du chiffrement symétrique : il n'est pas réaliste de supposer que toutes les clefs symétriques sont certifiées par un serveur de confiance.

Nous présentons ci-dessous plusieurs contre-exemples au théorème 9.1, dès lors que l'adversaire peut choisir ses clefs symétriques. Ces contre-exemples dépassent le cadre du théorème 9.1 et mettent en défaut les résultats de [BP04] et [JLM05] dans le cadre de l'abstraction symbolique du chiffrement symétrique.

Déchiffrement avec clef malhonnête. Supposons qu'un agent A envoie un message de la forme $k, c, \{c\}_{K_{ab}}$ où c est un message chiffré par la clef fraîche k, par exemple le message $c = \{n\}_k$ où n est un nonce frais. Cet envoi de message peut être représenté par le processus :

$$A = (\nu r)\overline{c_{\mathsf{out}}}(\mathsf{pair}(k, \mathsf{pair}(c, \mathsf{enc}(c, k_{ab}, r)))).\mathbf{0}$$

L'agent B attend une clef y ainsi qu'un message de la forme $\{\{b\}_y\}_{K_{ab}}$ où b est l'identité de B, et dans le cas où B reçoit le message attendu, il envoie un secret s (ou atteint un état « mauvais » pour le système).

$$\begin{aligned} A &\rightarrow B : \quad K, c, \{c\}_{K_{ab}} \\ B : K, \{\{B\}_K\}_{K_{ab}} &\rightarrow A : \quad s \end{aligned}$$

Le rôle de B peut être formalisé par le processus :

$$B = c_{\mathsf{in}}(z).\ \texttt{if}\ EQ(b, \mathsf{dec}(\mathsf{dec}(\pi_2(z), k_{ab}), \pi_1(z)))\ \texttt{then}\ \overline{c_{\mathsf{out}}}(s)\ \texttt{else}\ \mathbf{0}$$

Ainsi, dans le mode formel, le processus $(\nu k_{ab})(\nu s)A \| B$ n'émettra jamais s car l'agent A n'envoie pas de message de la forme attendue. Cependant, l'hypothèse de sécurité IND-CPA

[2] Définie page 74.

et IND-CTXT n'empêche *a priori* pas l'adversaire de forger une clef k tel que le déchiffrement de n'importe quelle suite de bits par k donne b. Il suffit alors à l'adversaire d'envoyer le message $\mathsf{pair}(k, \mathsf{enc}(c, k_{ab}, r))$ à B pour obtenir le secret s.

Cette attaque est due au fait que la sécurité des schémas de chiffrement n'apportent des garanties que pour les clefs de chiffrement *générées de manière valide*, c'est-à-dire en utilisant le générateur de clef associé au schéma de chiffrement. Considérons ainsi la construction suivante. Soit $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement IND-CPA et IND-CTXT. Nous pouvons construire le schéma de chiffrement $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ défini par $\mathcal{G}' = 0 \cdot \mathcal{G}$ (toutes les clefs honnêtes commencent par le bit 0), $\mathcal{E}'(m, i \cdot k) = \mathcal{E}(m, k)$ pour $i \in \{0, 1\}$ et où le déchiffrement $\mathcal{D}'$ est défini par :

- $\mathcal{D}'(c, k) = \mathcal{D}(c, k')$ si $k = 0 \cdot k'$;
- $\mathcal{D}'(c, k) = k'$ si $k = 1 \cdot k'$.

Le schéma de chiffrement $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ est toujours IND-CPA et IND-CTXT et permet de monter l'attaque décrite ci-dessus.

Pour rendre compte de ce type d'attaque, une possibilité (suggérée par M. Backes [Bac07]) est d'enrichir le pouvoir de l'attaquant symbolique en ajoutant une règle qui permet, étant donné un message chiffré c et un message m, de construire une clef k telle que le déchiffrement de c par m donne k. On peut ainsi ajouter un symbole fonctionnel $\mathsf{fakekey}$ d'arité 2 auquel on associe l'équation

$$\mathsf{dec}(x, \mathsf{fakekey}(x, y)) = y.$$

Cette nouvelle équation permettrait de retrouver symbolique l'attaque cryptographique mentionnée plus haut.

Déchiffrement caché. Cette solution se révèle cependant insuffisante pour couvrir tous les exemples suivants.

Ainsi, le message à déchiffrer pourrait ne pas être accessible de l'intrus. Considérons le protocole où l'agent A envoie le message $\mathsf{pair}(k, \mathsf{enc}(\mathsf{enc}(k', k, r'), k_{ab}, r))$ où k et k' sont des clefs fraîches. L'agent B peut calculer la clef k' et la renvoyer chiffrée par k_{ab} à A. Supposons que dans le cas où A reçoit son propre nom chiffré par k_{ab}, alors A envoie un secret s.

$$\begin{aligned} A &\to B : \quad K, \{\{K'\}_K\}_{K_{ab}} \\ B &\to A : \quad \{K'\}_{K_{ab}} \\ A : \{A\}_{K_{ab}} &\to B : \quad s \end{aligned}$$

Ce protocole peut être formellement décrit par les processus ci-dessous :

$$\begin{aligned} A &= (\nu r, r', k, k')\overline{c_{\mathsf{out}}}(\mathsf{pair}(k, \mathsf{enc}(\mathsf{enc}(k', k, r'), k_{ab}, r))).c_{\mathsf{in}}(z). \\ &\qquad \texttt{if } EQ(a, \mathsf{dec}(z, k_{ab})) \texttt{ then } \overline{c_{\mathsf{out}}}(s) \texttt{ else } \mathbf{0} \\ B &= (\nu r)c_{\mathsf{in}}(z).\overline{c_{\mathsf{out}}}(\mathsf{enc}(\mathsf{dec}(\mathsf{dec}(\pi_2(z), k_{ab}), \pi_1(z)), k_{ab}, r)) \end{aligned}$$

Ainsi, dans le mode formel, le processus $(\nu k_{ab})(\nu s)A \| B$ n'émettra jamais s car l'agent B n'envoie pas de message de la forme attendue par A. Pourtant, dans le modèle cryptographique, un adversaire peut à nouveau forger une clef k tel que le déchiffrement par k de tout message donne a.

Il est encore possible de refléter cette attaque dans le modèle symbolique en considérant une propriété plus forte pour la primitive $\mathsf{fakekey}$. Nous pouvons ainsi permettre à l'intrus de forger une clef k sans avoir à connaître le message à déchiffrer en ajoutant l'équation

$$\mathsf{dec}(x, \mathsf{fakekey}(y)) = y$$

où fakekey est cette fois une primitive d'arité 1.

Déchiffrement cachés et simultanés. Il est cependant possible de construire des attaques où il est nécessaire que le déchiffrement dépende du message à déchiffrer. Considérons ainsi le cas d'un protocole où l'agent A envoie p messages chiffrés $c_1, \ldots, c_p$ à B. Alors B chiffre tous les messages chiffrés ainsi qu'un nonce frais N_b, avec une clef K_{ab} partagée entre A et B. L'agent B envoie également p autres nonces frais $N_1, \ldots, N_p$, en clair. L'agent A renvoie alors le message de B, accompagné d'une clef K. Dans la dernière étape, l'agent B vérifie que le déchiffrement par K de chacun des messages chiffrés c_i donne le nonce N_i ; auquel cas il envoie un secret s.

$$\begin{array}{rcl} A & \to B: & c_1, \ldots, c_p \\ B & \to A: & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\ A & \to B: & \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, K \\ B: \{N_b, \{N_1\}_K, \ldots, \{N_p\}_K\}_{K_{ab}}, K & \to A: & s \end{array}$$

Ce protocole est décrit formellement par les processus :

$$\begin{aligned} A &= (\nu k)\overline{c_{\mathsf{out}}}(\mathsf{pair}(c_1, \mathsf{pair}(\ldots, c_p))).c_{\mathsf{in}}(z).\overline{c_{\mathsf{out}}}(\mathsf{pair}(\pi_1(z), k)).\mathbf{0} \\ B &= (\nu n_b, n_1, \ldots, n_p, r)c_{\mathsf{in}}(x).\overline{c_{\mathsf{out}}}(\mathsf{pair}(\mathsf{enc}(\mathsf{pair}(n_b, x), k_{ab}, r)\ , \mathsf{pair}(n_1, \mathsf{pair}(\ldots, n_p)))).c_{\mathsf{in}}(y). \\ &\quad \texttt{if}\ EQ(n_b, \pi_1((\mathsf{dec}(\pi_1(y), k_{ab}))) \wedge EQ(n_1, \mathsf{dec}(\pi_1(\pi_2(\mathsf{dec}(\pi_1(y), k_{ab}))), \pi_2(y))) \\ &\qquad \wedge EQ(n_2, \mathsf{dec}(\pi_1(\pi_2(\pi_2(\mathsf{dec}(\pi_1(y), k_{ab})))), \pi_2(y))) \\ &\qquad \wedge \cdots \wedge EQ(n_p, \mathsf{dec}(\pi_2(\cdots(\pi_2(\mathsf{dec}(\pi_1(y), k_{ab})))), \pi_2(y))) \\ &\quad \texttt{then}\ \overline{c_{\mathsf{out}}}(s)\ \texttt{else}\ \mathbf{0} \end{aligned}$$

Symboliquement, le processus $(\nu k_{ab})(\nu s)A \| B$ n'émettra jamais s car les nonces N_i sont générés *après* la réception des messages chiffrés c_i et le nonce N_b protège le protocole contre des attaques par rejeu. Cependant, dans le modèle cryptographique, un adversaire peut calculer une clef k à partir des messages chiffrés c_i et des nonces N_i, telle que le déchiffrement de c_i par k donne N_i. Pour cela, il faut considérer un schéma de chiffrement un peu différent de celui donné précédemment. Soit $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement IND-CPA et IND-CTXT et soit $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ le schéma défini par $\mathcal{G}' = 0 \cdot \mathcal{G}$ (toutes les clefs honnêtes commencent par le bit 0), $\mathcal{E}'(m, i \cdot k) = \mathcal{E}(m, k)$ pour $i \in \{0, 1\}$ et le déchiffrement $\mathcal{D}'$ est défini par :

- $\mathcal{D}'(c, k) = \mathcal{D}(c, k')$ si $k = 0 \cdot k'$;
- $\mathcal{D}'(c, k) = n$ si $k = 1 \cdot c_1, n_1, \cdots c, n \cdots c_p, n_p$;
- le chiffrement échoue sinon.

Le schéma de chiffrement $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ reste IND-CPA et IND-CTXT et permet à l'adversaire de choisir une clef pour monter l'attaque décrite ci-dessus.

Pour refléter cette attaque dans le modèle symbolique, il faudrait compliquer encore les équations à considérer. Ainsi, nous pourrions ajouter un symbole $\mathsf{fakekey}_p$ d'arité $2p$ pour tout $p \in \mathbb{N}$ ainsi que des équations de la forme

$$\mathsf{dec}(x_i, \mathsf{fakekey}_p(x_1, \ldots, x_p, y_1, \ldots, y_p)) = y_i$$

Chiffrement malhonnête. Cependant, ces équations ne permettraient toujours pas de capturer la (dernière) famille d'attaque que nous décrivons ci-dessous.

Considérons cette fois un protocole où l'agent B attend un message $\{N_a\}_{K_{ab}}$ de la part d'un agent A et une clef K de la part d'un agent C. L'agent B envoie alors à A le message $K, \{\{N_a\}_K\}_{K_{ab}}$ où le nonce N_a est chiffré par K puis par K_{ab}. Au cas où l'agent reçoit un message de la forme $K, \{\{N_a, N_a\}_K\}_{K_{ab}}$ où le nonce N_a est dupliqué, alors A émet un secret s.

$$\begin{aligned} A &\to B: \ \{N_a\}_{K_{ab}} \\ C &\to B: \ K \\ B &\to A: \ K, \{\{N_a\}_K\}_{K_{ab}} \\ A: k, \{\{N_a, N_a\}_K\}_{K_{ab}} &\to B: \ s \end{aligned}$$

Ce protocole peut être représenté par les processus ci-dessous.

$$\begin{aligned} A &= (\nu n_a, r)\overline{c_{\mathsf{out}}}(\mathsf{enc}(n_a, k_{ab}, r)).c_{\mathsf{in}}(z). \\ &\quad \texttt{if } EQ(\mathsf{pair}(n_a, n_a), \mathsf{dec}(\mathsf{dec}(\pi_2(z), k_{ab})), \pi_1(z)) \texttt{ then } \overline{c_{\mathsf{out}}}(s) \texttt{ else } \mathbf{0} \\ B &= (\nu r, r')c_{\mathsf{in}}(x).c_{\mathsf{in}}(y).\overline{c_{\mathsf{out}}}(\mathsf{enc}(\mathsf{enc}(\mathsf{dec}(x, k_{ab}), y, r'), k_{ab}, r)).\mathbf{0} \end{aligned}$$

Symboliquement, le processus $(\nu k_{ab})(\nu s)A \| B$ n'émettra jamais s alors que dans le modèle cryptographique, rien n'empêche *a priori* un adversaire de choisir une clef telle que le déchiffrement duplique le message obtenu. Nous pouvons construire les mêmes types de contre-exemples si l'agent A attend le message $\{\{N_a, N_a, N_a\}_K\}_{K_{ab}}$ (le nonce est répliqué trois fois) ou $\{\{N_a, A\}_K\}_{K_{ab}}$. Il est en fait possible de construire des attaques (et des schémas de chiffrement IND-CPA et IND-CTXT) où la clef définit une fonction à appliquer sur le message déchiffré et cette fonction peut varier pour chaque clef.

9.3 Perspectives

Une première motivation pour abstraire l'indistinguabilité cryptographique à l'aide de l'équivalence observationnelle est le traitement symbolique de la confidentialité calculatoire, définie à la partie 8.1.4. Supposons qu'un protocole soit représenté par le processus $P(x)$, paramétré par la valeur x du secret. Le processus $P(x)$ peut contenir des réplications et compositions parallèles pour modéliser plusieurs sessions. Dans le jeu définissant la confidentialité calculatoire, l'adversaire interagit d'abord avec le processus $\nu n_0, n_1 P(n_b)$ où $b \in \{0, 1\}$. Puis dans une deuxième phase, les deux valeurs n_0, n_1 sont révélées à l'adversaire. Il est possible de modéliser une notion plus forte, en révélant les deux valeurs dès le début. Ainsi, l'équivalence

$$\nu n_0, n_1 \overline{c_{\mathsf{out}}}(\mathsf{pair}(n_0, n_1)).(P(n_0)) \sim_o \nu n_0, n_1 \overline{c_{\mathsf{out}}}(\mathsf{pair}(n_0, n_1)).(P(n_1))$$

implique

$$[\![\nu n_0, n_1 \overline{c_{\mathsf{out}}}(\mathsf{pair}(n_0, n_1)).(P(n_0))]\!] \approx [\![\nu n_0, n_1 \overline{c_{\mathsf{out}}}(\mathsf{pair}(n_0, n_1)).(P(n_1))]\!]$$

d'après le théorème 9.1 et donc la confidentialité calculatoire de x dans P. Il s'agit cependant d'une propriété strictement plus forte que la confidentialité calculatoire puisque l'adversaire peut tirer parti de la connaissance *a priori* des valeurs n_0, n_1 pour distinguer les deux processus. Pour modéliser la confidentialité calculatoire de manière exacte, il faudrait ajouter un opérateur de *phase* permettant l'arrêt brutal de tous les processus en cours et le passage au

processus suivant. Si l'opérateur de phase est noté $\cdot$, la confidentialité calculatoire s'exprime alors exactement par la propriété

$$[\![\nu n_0, n_1(P(n_0)).\overline{c_{\mathsf{out}}}(\mathsf{pair}(n_0, n_1))]\!] \approx [\![\nu n_0, n_1(P(n_1)).\overline{c_{\mathsf{out}}}(\mathsf{pair}(n_0, n_1))]\!]$$

Comme notre résultat de correction de l'équivalence observationnelle ne semble pas dépendre de la structure de contrôle des processus, nous devrions pouvoir facilement l'étendre aux processus avec opérateur de phase. Cela nous permettrait ainsi de prouver symboliquement le secret calculatoire. Cet opérateur serait également utile pour définir certains protocoles comme les protocoles de vote électronique, qui comportent explicitement plusieurs étapes (enregistrement, vote et comptage). L'opérateur de phase permettrait de modéliser la synchronisation des processus à chaque étape.

Les résultats de cette partie font surgir de nouveaux problèmes de décidabilité. Ainsi, le théorème 9.1 s'applique à des processus avec prédicats $(M, EQ, \mathsf{P}_{\mathsf{samekey}}, EL)$. Il faut donc rétablir la décidabilité de l'équivalence observationnelle (partie 7.3.3) pour des processus (simples) comportant de tels prédicats. Cela nécessite en premier lieu de revisiter la procédure de décision proposée par Mathieu Baudet [Bau05, Bau07]. Le prédicat EQ est celui habituellement considéré, il ne demandera donc aucun travail supplémentaire. Le prédicat M, garantissant l'absence de destructeurs, est inhabituel dans les résultats traitant des théories équationnelles. Cependant, puisque ce prédicat permet un rapprochement avec les résultats où les destructeurs sont implicites, il est raisonnable d'espérer pouvoir traiter le prédicat M assez facilement. Par contre, le prédicat EL qui permet de comparer les longueurs symboliques de messages pourrait conduire plus rapidement à des résultats d'indécidabilité puisqu'il permet de coder un fragment de l'arithmétique. Une première approche pourra consister à proposer une sur-approximation de l'équivalence observationnelle. Une autre solution est de restreindre la classe des protocoles considérée de manière à assurer qu'un nombre fini de longueurs différentes de messages pourront être utilisées dans les sessions honnêtes.

D'autre part, nous avons proposé (partie 9.2.3) plusieurs exemples démontrant les problèmes causés par des clefs choisies de manière malhonnête. Cependant, la solution consistant à supposer une infrastructure de confiance pour les clefs symétriques n'est pas réaliste. Une autre solution consisterait à restreindre à nouveau la classe des protocoles aux protocoles qui assurent la bonne distribution des clefs avant leur utilisation pour chiffrer ou déchiffrer. Cette solution élimine en particulier tous les contre-exemples mentionnés à la partie 9.2.3. Il faudra alors s'assurer que la présence de clefs malhonnêtes dans les sessions qui comportent au moins un agent corrompu n'interfère pas avec les sessions honnêtes.

De nombreuses autres perspectives liées à ce chapitre sont présentées au chapitre 10.

CHAPITRE 10

Perspectives

Dans ce mémoire, nous avons présenté nos travaux de recherche les plus récents autour de deux axes principaux. D'une part, nous avons proposé des procédures de décision pour l'analyse formelle des protocoles cryptographiques, adaptées à différents scénarios : attaquant actif ou passif, nombre de sessions borné ou non, différentes propriétés et primitives cryptographiques. D'autre part, nous avons étudié sous quelles hypothèses les preuves obtenues dans les modèles symboliques donnent des garanties dans les modèles plus précis que sont les modèles cryptographiques. À la fin de chaque chapitre, nous avons décrit les développements naturels des résultats présentés.

L'objet de cette partie est de reprendre et de développer les principales directions de recherche futures. Elles s'articulent autour de trois thèmes principaux. D'une part, nous projetons d'analyser de nouvelles familles de protocoles comme les protocoles de routage sécurisé ou les interfaces de programmation dédiées aux modules matériels de sécurité (partie 10.1). D'autre part, les résultats concernant la correction des modèles symboliques vis-à-vis des modèles calculatoires sont prometteurs mais encore insatisfaisants. Nous en discutons les perspectives dans la partie 10.2 de ce chapitre. Enfin, le nombre de résultats obtenus sur l'analyse des protocoles cryptographiques est désormais suffisant pour tenter de les composer entre eux. C'est l'axe de recherche que nous présentons dans la partie 10.3.

10.1 Nouveaux protocoles

Les résultats de décidabilité et d'indécidabilité sont désormais bien compris pour les protocoles classiques comme ceux de la bibliothèque de protocoles présentée par Clark&Jacob [CJ97]. Nous avons ainsi rédigé un état de l'art sur le sujet *[CDL06]*. Cependant, de nouvelles familles de protocoles sont apparues comme les protocoles de vote électronique, les protocoles de signatures de contrat, les protocoles de routage sécurisé ou les interfaces de programmation dédiées aux modules matériels de sécurité. Nous avons commencé à aborder les problématiques soulevées par ces nouvelles applications au cours de ce mémoire. Ainsi le chapitre 4 aborde la modélisation et l'analyse des interfaces de programmation dédiées aux modules matériels de sécurité. D'autre part, l'étude de l'équivalence statique (chapitre 6) et de l'équivalence observationelle (chapitre 7) est motivée par les définitions de nouvelles propriétés à l'aide d'équivalence comme l'anonymat dans le vote électronique [KR05, DKR09b]. Nous consacrons la première partie de ce chapitre aux développements nécessaires pour traiter ces nouvelles familles de

protocoles.

10.1.1 Interfaces de programmation

Nous avons vu (partie 4.3 du chapitre 4) que les interfaces de programmation (API) dédiées aux modules matériels de sécurité (HSM) ont pour but de garantir la sécurité des données même lorsque le module de sécurité est connecté à un environnement hostile comme un ordinateur infecté par un virus par exemple.

Les API comportent différentes commandes qui ont la particularité d'être *sans mémoire* : chaque commande est exécutée indépendamment des commandes précédentes. Nous avons vu comment cette particularité rend la modélisation en clauses de Horn particulièrement pertinente. Nous avons proposé plusieurs classes décidables *[CKS07, CDS07a]* permettant d'analyser les API. Nous avons également proposé une construction générique permettant d'implémenter la majorité des protocoles à chiffrement symétrique *[CS09a]*.

Un plus grand nombre de primitives. Cependant, les résultats de décidabilité actuels ne prennent en compte que trois primitives : le chiffrement symétrique et l'opérateur de « ou exclusif » ainsi que la concaténation. Même si une infrastructure de clefs asymétriques est plus lourde et coûteuse à mettre en place, certaines applications requièrent du chiffrement asymétrique ou des signatures. Les API font également une utilisation importante des fonctions de hachage. Il est donc nécessaire de généraliser les résultats existants à des primitives plus nombreuses comme le chiffrement asymétrique, les macs (Message authentication code), le hachage ou les signatures. Pour l'API générique que nous avons proposée dans le cadre du chiffrement symétrique (partie 4.3.4), prendre en compte le chiffrement asymétrique nécessite en premier lieu d'adapter la notion de confiance mise en place dans les APIs. En effet, dans le cadre du chiffrement symétrique, lorsqu'un agent reçoit un message chiffré par une clef k appartenant à un groupe d'utilisateurs S, l'agent peut être sûr que les données ont été chiffrées par un des utilisateurs de S et sont donc « aussi fiables » que le niveau de fiabilité des agents de S. À l'inverse, une donnée chiffrée par une donnée publique $\mathsf{pub}(a)$ peut provenir de n'importe quelle source, en particulier d'un intrus. La transmission de la confiance demanderait par exemple une étape d'authentification.

Adaptation du modèle de l'intrus. D'autre part, l'étude des API revient à étudier des protocoles cryptographiques dans le cadre d'un modèle de l'attaquant renforcé où une partie des calculs intermédiaires effectués par les agents honnêtes (lors du chiffrement et du déchiffrement des données) peut être interceptée par un attaquant. Comme le suggérent David Basin et Cas Cremers [BC09], un angle d'attaque possible consiste à utiliser les outils existants pour les protocoles cryptographiques, en ajoutant des envois de messages représentant explicitement les pertes possibles d'information de la part des agents honnêtes. La difficulté de ce codage réside dans l'identification des données qui doivent être transmises à l'intrus pour refléter les attaques éventuelles liées à l'infection d'un ordinateur. Les données à transmettre dépendent de l'API considérée (mieux l'API est conçue, moins les données sont perdues). On peut donc imaginer une transformation automatique qui, étant donnée une API et un protocole, calcule les règles à ajouter pour analyser ce protocole implémenté à l'aide de l'API pour la gestion des données sensibles.

Prise en compte de la cryptographie. Un autre aspect concerne la prise en compte des attaques liées à la cryptographie. En effet, les API de sécurité utilisent des primitives de chiffrement assez simples qui ne satisfont pas les plus hauts niveaux de sécurité actuels. Ainsi, les schémas de chiffrement utilisés sont souvent IND-CPA mais non IND-CCA. Les résultats de correction développés aux chapitres 8 et 9 ne peuvent donc pas s'appliquer pour obtenir des résultats au niveau calculatoire. De nouveaux types d'attaques sont en fait possibles. Ces attaques font souvent intervenir des aspects *quantitatifs*. Par exemple, un attaquant peut deviner une partie des clefs ou essayer des clefs générées au hasard mais cela lui « coûte » du temps de calcul. Suivant le temps de calcul associé à une attaque et le niveau de sécurité souhaitée, l'attaque peut être considérée ou non comme réaliste. Nous avons ainsi proposé une modélisation complètement symbolique des techniques de « Key Conjuring » *[CDS07a]* (partie 4.3.3). Cependant, pour aller plus loin, il semble nécessaire d'intégrer des aspects quantitatifs dans les modèles formels de sécurité, ce qui est encore très peu développé à l'heure actuelle. Nous pensons donc proposer de nouveaux types de modèles, qui pourront également être utiles à d'autres types d'applications.

Propriétés. Enfin, les propriétés analysées jusqu'ici se limitent principalement à des notions de secret. Les nouvelles applications en cours de développement comme l'utilisation d'API de sécurité dans les voitures [RH07] demandent la vérification de propriétés plus complexes comme le respect de l'anonymat des utilisateurs (pour éviter qu'une voiture puisse être tracée) ou l'évaluation de la distance parcourue (sur des portions d'autoroute payante par exemple). L'identification et la formalisation des propriétés les plus importantes est la première étape à réaliser.

10.1.2 Protocoles de routage sécurisés.

Pour transmettre un message entre deux points du réseau, les données sont transmises de nœuds voisins en nœuds voisins. Les protocoles de routage ont comme objectif d'établir une « route », c'est-à-dire un chemin entre deux points du réseau tel que chaque arête du chemin relie des nœuds voisins. La première étape d'une attaque consiste souvent à faire accepter à un nœud honnête des routes malhonnêtes qui, soit empêchent le nœud de communiquer avec une partie de ses correspondants, soit forcent les communications à passer par un nœud contrôlé par l'attaquant. Pour contrer ce type d'attaque, plusieurs protocoles de routage dits sécurisés ont été proposés [PR99, HPJ05, BV04, GZA02, SKY05].

Ces protocoles amènent des problématiques nouvelles. Ainsi, nous avons vu (partie 3.3.2) que les protocoles de routage font des tests récursifs, pour tester la validité d'une route par exemple. Ces protocoles demandent également de modéliser un attaquant plus faible qui ne contrôle non plus toutes les communications du réseau mais seulement (en partie) celles de ses voisins. Cela revient à permettre moins d'exécutions par rapport au modèle classique d'un attaquant dit Dolev-Yao. Aussi il semble possible d'aborder ce nouveau modèle avec des techniques existantes comme les systèmes de contraintes (chapitre 3).

D'autre part, pour modéliser certaines attaques comme les « rushing attacks » [HPJ03], il est nécessaire de prendre en compte les temps de transmission de messages [SSBC09]. Il faut alors trouver un compromis entre la précision de la modélisation et la possibilité de mettre en œuvre des procédures automatiques de décision.

Plutôt que de chercher directement à identifier de nouvelles classes décidables pertinentes pour les protocoles de routage sécurisé, il semble pertinent de mettre en place des résul-

tats de simplification. Ainsi, la présence de deux nœuds malhonnêtes permet plus d'attaques que la présence d'un seul nœud malhonnête. Ces attaques sont appelées attaques par « tunnel » [SDL+02]. Mais est-il utile de considérer plus de deux nœuds malhonnêtes ? De manière similaire, est-il nécessaire de considérer un nombre arbitraire de nœuds (honnêtes) pour détecter des attaques ? Existe-t-il des configurations types du réseau qui permettent à coup sûr de détecter toutes les attaques possibles ? Ce sont des questions qui, à ma connaissance, restent ouvertes.

10.1.3 Protocoles de vote.

Le vote électronique est une technique prometteuse pour permettre de recueillir les résultats d'une élection de manière sûre, rapide et pratique. Cela permet en particulier d'éviter la tâche fastidieuse du dépouillement. Cela permet également d'éviter aux électeurs de se déplacer. Le vote électronique peut être utilisé pour un type varié d'élections, allant de petits comités à des élections nationales, en passant par des élections syndicales par exemple. En France, la première utilisation du vote à travers Internet date de 2003 pour permettre aux citoyens français résidants aux États-Unis d'élire leurs représentants à l'assemblée. Depuis, le vote électronique et les machines à voter sont largement déployées malgré les réserves qui ont pu être soulevées. Ainsi, des travaux récents [FHF06] ont permis l'étude de la sécurité de la machine Diebold AccuVote-TS, à la fois sur des aspects logiciels et matériels. Les résultats de l'étude ont montré que cette machine est vulnérable à de nombreuses attaques sérieuses. Il est donc crucial d'analyser ces protocoles. Les garanties souhaitées sont nombreuses. Stéphanie Delaune, Steve Kremer et Mark Ryan [DKR09b] ont formalisé plusieurs propriétés comme l'anonymat, la propriété de sans reçu ou la résistance à la coercition à l'aide d'équivalence observationelle. Ainsi, le secret d'un vote pour un protocole avec deux votants A et B s'exprime par

$$A({}^{a}/_{x})\|B({}^{b}/_{x}) \sim_o A({}^{b}/_{x})\|B({}^{a}/_{x}).$$

Un attaquant ne doit pas pouvoir faire la différence du cas où un agent A vote a et B vote b du cas inverse où A vote b et B vote a.

Nous avons également vu au chapitre 6 que les protocoles de vote utilisent des primitives particulières comme les signatures en aveugle, le rechiffrement, les preuves à vérificateur désigné ou les fonctions d'engagement partiel.

Des résultats ont été obtenus d'une part pour le traitement de l'équivalence observationelle *[CD09a]*, [BAF05, BBN04, BN05, JPVB08, DKR07, DKR09a] et d'autre part pour le traitement des théories équationelles reflétant les primitives utilisées dans le vote électronique *[BBRC09]*, [CDK09]. Mais à l'heure actuelle, rien ne permet de combiner ces deux résultats, c'est-à-dire traiter l'équivalence observationelle dans le cas des théories liées aux protocoles de vote. Les protocoles de vote comme FOO 92 [FOO92] ou comme le protocole de Lee *et al* [LBD+03] constituent des études de cas réalistes et importantes qui motivent le développement de nouvelles techniques qui seront utiles pour d'autres familles de protocoles. Cet axe de recherche fait partie des axes de recherche du projet ANR SeSur AVOTÉ[1].

[1] `http://www.lsv.ens-cachan.fr/Projects/anr-avote/`

10.2 Développement du lien avec les modèles calculatoires

Les modèles symboliques utilisés pour analyser les protocoles font complètement abstraction de la cryptographie. Les chapitres 8 et 9 montrent qu'il est possible d'obtenir des garanties en tenant compte de la cryptographie, à l'aide des modèles symboliques et en considérant des primitives cryptographiques suffisamment robustes. Nous avons récemment rédigé un article résumant l'état de l'art actuel sur le sujet *[CKW09]*. Néanmoins, les résultats se limitent pour le moment à un nombre encore restreint de primitives cryptographiques et toutes les hypothèses ne sont pas satisfaisantes (notion de sécurité très forte, infrastructure de clefs particulière, *etc.*).

10.2.1 Généralisation des résultats existants

Une première étape consiste à généraliser les résultats déjà obtenus. Ainsi, nous avons vu au chapitre 9 (partie 9.3) qu'il serait possible de traiter un plus grand nombre de propriétés cryptographiques à l'aide de l'équivalence observationelle, à condition d'ajouter à l'algèbre de processus un opérateur de « phase » qui permet de resynchroniser les processus. Il semble également possible d'améliorer l'hypothèse d'une infrastructure de confiance pour les clefs symétriques en restreignant la classe de protocoles considérée. Une autre possibilité pour traiter le chiffrement symétrique est de considérer une hypothèse cryptographique plus forte sur le chiffrement. Ainsi, la construction proposée par Marc Fischlin [Fis99] empêche de choisir une clef de déchiffrement après avoir produit un message chiffré, ce qui semble écarter les contre-exemples décrits à la partie 9.2.3.

Plus généralement, les nouvelles familles de protocoles évoquées à la partie 10.1 apportent de nouvelles primitives cryptographiques à intégrer aux résultats de correction. Ainsi, Yusuke Kawamoto *et al.* a montré [KSH08] comment les signatures en aveugle peuvent être abstraites dans les modèles symboliques, de manière correcte vis-à-vis des modèles cryptographiques. Michael Backes *et al.* a également défini [BMU08] une représentation symbolique pour les protocoles de preuve à divulgation nulle, en a montré la correction vis-à-vis des modèles cryptographiques [BU08] et a proposé une procédure de décision à l'aide de systèmes de typage [BHM08]. Les primitives comme le rechiffrement, les preuves à vérificateur désigné ou les fonctions d'engagement partiel n'ont par contre encore jamais été abordées dans ce contexte. Il faudra alors mettre au point un modèle symbolique correct vis-à-vis des hypothèses cryptographiques existantes, en s'appuyant sur les modélisations symboliques déjà proposées.

Chaque résultat de correction traite les primitives une à une. Un résultat reprenant toutes les primitives en même temps serait pourtant très souhaitable pour traiter des études de cas réalistes mais demanderait une preuve complexe. Plutôt que traiter les primitives une à une ou refaire les preuves pour les traiter ensemble, il semble possible de concevoir des résultats de façon modulaire, de manière à combiner gratuitement les primitives. Cet aspect sera développé dans la dernière partie de ce chapitre.

10.2.2 Retour vers les modèles symboliques

Pour garantir des propriétés cryptographiques à l'aide des modèles symboliques, il est parfois nécessaire d'enrichir ceux-ci. Ainsi le traitement des signatures et du chiffrement asymétrique a demandé, d'une part, d'introduire un opérateur explicite pour les signatures (rarement traité par les outils) et d'autre part, de modéliser explicitement l'aléa utilisé en ajoutant un

troisième argument aux opérateurs de chiffrement et signature. Nous avons montré *[CHW07]* qu'il est possible de traiter un tel modèle symbolique enrichi à l'aide des outils existants.

Nous avons également vu à la partie 10.1 que les prédicats ajoutés pour le traitement de l'équivalence observationelle comme le prédicat M, qui s'assure qu'un terme ne contient pas de destructeurs, ou le prédicat EL, qui s'assure que deux termes ont la même longueur, sont plus problématiques et demandent de redéfinir des procédures de décision adaptées. Faire coïncider les modèles symboliques avec les modèles cryptographiques demande un bon compromis entre l'enrichissement des modèles symboliques (qui doivent garder leur relative simplicité pour permettre des preuves automatiques ou automatisables) et le renforcement des hypothèses cryptographiques sur les primitives (qui doivent cependant rester implémentables et réalistes).

10.3 Modularité

Les résultats de décidabilité obtenus pour la vérification des protocoles cryptographiques ainsi que les résultats de correction des modèles symboliques vis-à-vis des modèles calculatoires sont désormais en nombre conséquent. Chaque résultat traite en général une primitive ou une propriété particulière pour simplifier la présentation et surtout les preuves. Pourtant, ce domaine de recherche semble avoir désormais atteint un niveau de maturité suffisant pour que l'on cherche à combiner les résultats entre eux. Nous proposons plusieurs pistes pour généraliser le développement modulaire de l'analyse des protocoles.

10.3.1 Modularité dans les modèles symboliques

Dans le cadre des modèles symboliques, et pour quelques problématiques précises, nous avons obtenu des premiers résultats de composition. Nous avons par exemple prouvé que la décidabilité de la déduction et de l'équivalence statique pour des théories disjointes implique la décidabilité pour l'union des théories *[ACD07a]*. D'autre part, nous avons montré sous quelles hypothèses un protocole sûr peut être utilisé simultanément avec d'autres protocoles partageant des clefs *[CDD07a]*. Nous pensons que ces deux types de résultats peuvent être développés.

10.3.1.1 Composition de primitives cryptographiques

Notre résultat de combinaison *[ACD07a]* pour la décidabilité de la déduction et de l'équivalence observationelle s'applique uniquement au cas passif et pour des théories disjointes. Il semble possible de traiter des théories équationelles non disjointes, en considérant par exemple des hiérarchies comme proposé par Yannick Chevalier et Michael Rusinowitch [CR08]. Cela permettrait par exemple de traiter l'exponentiation modulaire combinée au chiffrement.

D'autre part, les résultats de décidabilité obtenus jusqu'ici sont particuliers à quelques primitives cryptographiques ou, au mieux, à des classes de primitives comme les primitives représentées par théories sous-terme convergentes [DJ04, Bau05]. Une question naturelle consiste à se demander s'il est possible de composer de telles théories. Si des propriétés telles que le secret et l'authentification sont décidables pour deux groupes E_1 et E_2 de primitives, qu'en est-t-il de la décidabilité pour l'union $E_1 \cup E_2$ des primitives ? Un premier élément de réponse a été apporté dans [CR08] dans le cadre d'un nombre borné de sessions et pour des propriétés d'accessibilité. La question n'a pas été abordée dans le cadre d'un nombre non borné de sessions ni pour l'équivalence observationelle.

10.3.1.2 Abstraction et raffinement de protocoles

Les protocoles utilisés en pratique sont plus en plus complexes et il devient difficile de les vérifier d'un bloc, même dans les modèles symboliques. Nous avons montré *[CDD07a]* qu'il est possible de vérifier un protocole indépendemment de son environnement, dès lors que les protocoles sont tagués : si P est sûr, alors il en est de même pour $P \mid Q$ dès lors que P et Q sont tagués. Plus formellement, nous avons montré l'implication suivante :

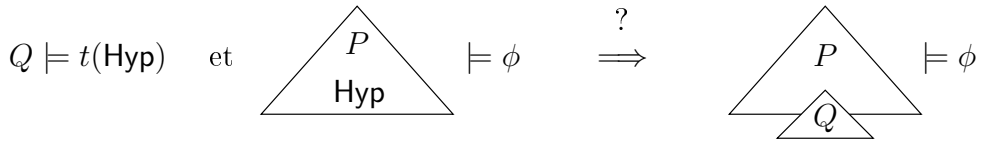
$$\nu k P \models \phi \;\Rightarrow\; \forall Q,\;\; \nu k(P \mid Q) \models \phi$$

Nous avons évoqué (partie 5.3) la possibilité d'obtenir un résultat équivalence dans le cadre de l'équivalence observationelle, qui permet de modéliser des propriétés plus générales. Intuitivement, nous aimerions obtenir l'implication

$$\nu k P_1 \sim_o \nu k P_2 \;\stackrel{?}{\Rightarrow}\; \nu k(P_1 \mid Q) \sim_o \nu k(P_2 \mid Q)$$

Nous pensons qu'il est possible d'aller plus loin en étudiant la composition *modulaire* : si un protocole est sûr sous l'hypothèse qu'il utilise (par exemple) un canal confidentiel entre deux agents, peut-on utiliser n'importe quel (sous)-protocole pour implémenter le canal confidentiel ? Plus généralement, nous souhaitons explorer le raffinement sûr de protocoles afin de permettre un développement modulaire, préservant la sécurité des protocoles.

Ainsi, supposons que nous ayons montré la sécurité du protocole P pour une propriété ϕ, c'est-à-dire, $P \models \phi$, sous certaines hypothèses Hyp concernant l'implémentation (canaux confidentiels, fiables, authentifiés, *etc.*). Quelle est la propriété $t(\mathsf{Hyp})$ à assurer sur le protocole Q implémentant Hyp pour que le protocole P combiné à Q satisfasse toujours ϕ ?



En d'autres termes, il s'agit d'étudier l'abstraction et le raffinement de protocoles, pour des propriétés de sécurité.

10.3.2 Développement modulaire des résultats de correction

Dans les différents chapitres de ce mémoire, nous avons vu que les traces d'exécutions concrètes d'un adversaire sont toutes représentées par les traces d'exécutions symboliques (sauf pour un nombre négligeable d'entre elles et sous réserve que les primitives cryptographiques satisfassent certaines hypothèses). Ce résultat, appelé « mapping lemma », a été établi pour la concaténation, le chiffrement asymétrique et les signatures (théorème 8.1), pour la concaténation, le chiffrement asymétrique et les fonctions de hachage (théorème 8.3) ainsi que pour la concaténation et le chiffrement symétrique (partie 9.2.2.3). Montrer ce résultat pour les cinq primitives en même temps paraît naturel mais demanderait une preuve très complexe. Comme les primitives paraissent assez indépendantes, une question naturelle se pose : si le « mapping lemma » est établi pour deux groupes de primitives distincts, est-ce que le « mapping lemma » est immédiatement vrai pour l'union des groupes de primitives ?

Il semble difficile d'obtenir directement un tel résultat de composition. Aussi, en collaboration avec Bogdan Warinschi, nous pensons qu'il est possible de mettre au point une notion

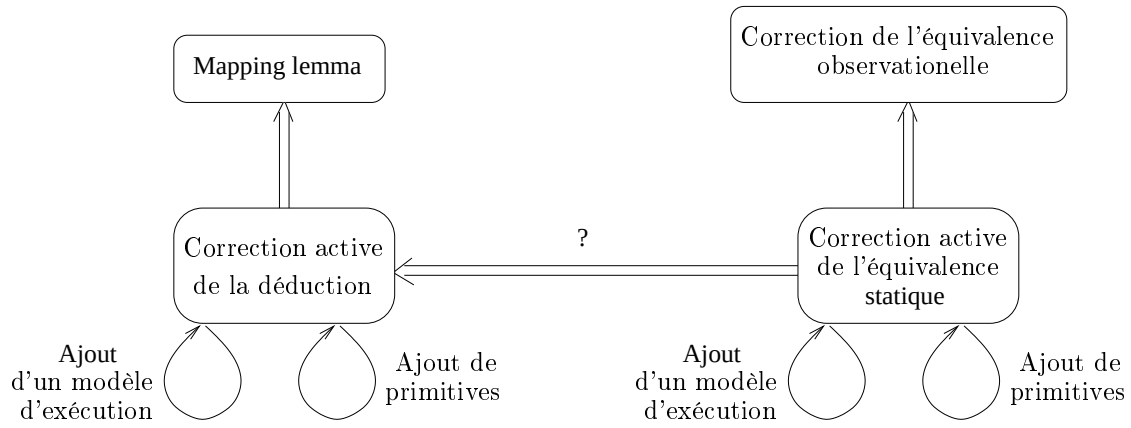


FIGURE 10.1 - *Correction de la déduction et de l'équivalence dans le cas actif.*

de *correction active de la déduction*, appelée $\vdash_E$-active soundness, pour une théorie équationelle E, inspirée de l'équivalence statique adaptative [KM07] et qui s'abstrait du modèle d'exécution des protocoles. La correction active de la déduction pour une théorie E permettrait de déduire immédiatement le « mapping lemma » pour cette théorie. L'avantage de cette notion est multiple : elle permettrait d'une part d'enrichir le modèle d'exécution (avec l'ajout de conditionnelle et de boucles par exemples) et d'autre part d'enrichir facilement les primitives (avec l'ajout de listes, de tableaux et d'autres primitives cryptographiques). Dans un premier temps, nous espérons pouvoir montrer que la correction active de la déduction pour une théorie E implique directement la correction active de la déduction pour le chiffrement symétrique et asymétrique ajoutés à la théorie E. Dans un deuxième temps, nous espérons pouvoir obtenir plus généralement un résultat de la forme

$$\vdash_{E_1}\text{-active soundness} + \vdash_{E_2}\text{-active soundness} \quad \stackrel{?}{\Rightarrow} \quad \vdash_{E_1 \cup E_2}\text{-active soundness}$$

Pour l'équivalence observationelle, il semble possible de définir une notion équivalente, appelée *correction active de l'équivalence* qui impliquerait la correction de l'équivalence observationelle. À nouveau, la correction active de l'équivalence serait beaucoup plus facilement composable. La figure 10.1 résume l'ensemble des propriétés que nous pensons pouvoir obtenir :

- Une notion de correction active de la déduction qui implique le « mapping lemma », qui permet de composer facilement les primitives cryptographiques et pour laquelle on peut facilement modifier le modèle d'exécutions.
- Une notion de correction active de l'équivalence qui implique la correction de l'équivalence observationelle et pour laquelle on pourrait ajouter de la même manière primitives cryptographiques et modèle d'exécutions.
- Il serait alors naturel d'étudier le lien entre les deux notions. Il est probable qu'elles soient incomparables en général mais que la correction active de l'équivalence implique la correction active de la déduction en présence de fonction de hachage, comme ce que nous avons montré dans le cas passif *[BCK05]* (partie 9.1).

Références bibliographiques

[AB05] M. Abadi and B. Blanchet. Computer-Assisted Verification of a Protocol for Certified Email. *Science of Computer Programming*, 58(1–2) :3–27, October 2005. Special issue SAS'03.

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, Edinburgh, Scotland, 2005.

[ABF04] M. Abadi, B. Blanchet, and C. Fournet. Just Fast Keying in the Pi Calculus. In D. Schmidt, editor, *Programming Languages and Systems : Proceedings of the 13th European Symposium on Programming (ESOP'04)*, volume 2986 of *Lecture Notes on Computer Science*, pages 340–354. Springer Verlag, Barcelona, Spain, March 2004.

[ABF07] M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security (TISSEC)*, 10(3) :1–59, July 2007.

[AC02] R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Proc. of the 13th International Conference on Concurrency Theory (CONCUR'02)*, LNCS, pages 499–514. Springer Verlag, 2002.

[AC05] A. Armando and L. Compagna. An optimized intruder model for sat-based model-checking of security protocols. In *Proceedings of the Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA 2004)*, volume 125 of *ENTCS*, pages 91–108. 2005.

[AD07] M. Arapinis and M. Duflot. Bounding messages for free in security protocols. In V. Arvind and S. Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 376–387. Springer, New Delhi, India, December 2007. doi :10.1007/978-3-540-77050-3_31.

[ADK08] M. Arapinis, S. Delaune, and S. Kremer. From one session to many : Dynamic tags for security protocols. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 128–142. Springer, Doha, Qatar, November 2008. doi :10.1007/978-3-540-89439-1_9.

[AF01] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. January 2001.

[AG97] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols : The spi calculus. In *Proc. of the 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

[AG98] M. Abadi and A. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4) :267–303, 1998.

[AL00] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proc. of the 12th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *LNCS*, pages 380–394. 2000.

[ALV02] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1) :695–740, 2002.

[AN96] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1) :6–15, 1996.

[AR00] M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proc. of the International Conference on Theoretical Computer Science (IFIP TCS2000)*, pages 3–22. August 2000.

[AR02] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 2 :103–127, 2002.

[AR07] M. Abadi and P. Rogaway. Reconciling two views of cryptography : the computational soundness of formal encryption. *J. Cryptology*, 2007.

[AW04] M. Abdalla and B. Warinschi. On the minimal assumptions of group signature schemes. In *6th International Conference on Information and Communication Security*, pages 1–13. 2004.

[BA01] M. Bond and R. Anderson. API level attacks on embedded systems. *IEEE Computer Magazine*, pages 67–75, October 2001.

[Bac07] M. Backes. Private communication, 2007.

[BAF05] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 331–340. IEEE Computer Society, June 2005.

[BAF08] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1) :3–51, 2008.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. of the Royal Society*, volume 426 of *Series A*, pages 233–271. 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[Bau05] M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, Alexandria, Virginia, USA, November 2005. doi :10.1145/1102125.

[Bau07] M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, January 2007.

[BB03] M. Boreale and M. G. Buscemi. Symbolic analysis of crypto-protocols based on modular exponentiation. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, volume 2747 of *LNCS*, pages 269–278. Springer-Verlag, Bratislava (Slovak Republic), 2003.

[BBM00] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting : Security proofs and improvements. In *Proc. of Eurocrypt'00*, volume 1807 of *LNCS*, pages 259–274. 2000.

[BBN04] J. Borgström, S. Briais, and U. Nestmann. Symbolic bisimulations in the spi calculus. In *Proc. of the 15th International Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *LNCS*, pages 161 – 176. Springer-Verlag, London, UK, 2004.

[BC08] B. Blanchet and A. Chaudhuri. Automated formal analysis of a protocol for secure file sharing on untrusted storage. In *Proceedings of the 29th IEEE Symposium on Security and Privacy (S&P'08)*, pages 417–431. IEEE, Oakland, CA, 2008.

[BC09] D. Basin and C. Cremers. From dolev-yao to strong adaptive corruption : Analyzing security in the presence of compromising adversaries. Cryptology ePrint Archive, Report 2009/079, 2009. `http://eprint.iacr.org/`.

[BCD07] S. Bursuc, H. Comon-Lundh, and S. Delaune. Associative-commutative deducibility constraints. In W. Thomas and P. Weil, editors, *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS'07)*, volume 4393 of *Lecture Notes in Computer Science*, pages 634–645. Springer, Aachen, Germany, February 2007. doi :10.1007/978-3-540-70918-3_54.

[BCJS02] F. Butler, I. Cervesato, A. Jaggard, and A. Scedrov. A formal analysis of some properties of kerberos 5 using MSR. In S. Schneider, editor, *Proc. of 15th IEEE Computer Security Foundations Workshop (CSFW'15)*, pages 175–190. IEEE Computer Society Press, June 2002.

[BCK98] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *STOC'98 : Proc. of the 30th ACM Symposium on Theory of Computing*, pages 419–428. 1998. ISBN 0-89791-962-9. doi :http://doi.acm.org/10.1145/276698.276854.

[BCL08] V. Bernat and H. Comon-Lundh. Normal proofs in intruder theories. In *Advances in Computer Science - ASIAN 2006*, volume 4435 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2008.

[BDNP99] M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Logic in Computer Science*, pages 157–166. 1999.

[BHK07] Y. Boichut, P.-C. Héam, and O. Kouchnarenko. Tree Automata for Detecting Attacks on Protocols with Algebraic Cryptographic Primitives. In *INFINITY'07, Int. Ws. on Verification of Infinite-State Systems, joint to CONCUR'07*, pages 44–53. Lisboa, Portugal, September 2007.

[BHKO04] Y. Boichut, P.-C. Heam, O. Kouchnarenko, and F. Oehl. Improvements on the Genet and Klay Technique to Automatically Verify Security Protocols. In *Proc. Int. Workshop on Automated Verification of Infinite-State Systems (AVIS'2004), joint to ETAPS'04*, pages 1–11. Barcelona, Spain, 2004. The final version will be published in EN in Theoretical Computer Science, Elsevier.

[BHM08] M. Backes, C. Hritcu, and M. Maffei. Type-checking zero-knowledge. In *Proceedings of 2008 ACM Conference on Computer and Communication Security (CCS'08*. 2008.

[Bla01] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, June 2001.

[Bla02] B. Blanchet. From secrecy to authenticity in security protocols. In *9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *LNCS*, pages 242–259. Springer-Verlag, September 2002.

[Bla04] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *IEEE Symposium on Security and Privacy (SP'04)*, pages 86–100. Oakland, California, May 2004.

[Bla05] B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *20th International Conference on Automated Deduction (CADE-20)*. Tallinn, Estonia, July 2005.

[BLMW07] E. Bresson, Y. Lakhnech, L. Mazaré, and B. Warinschi. A generalization of ddh with applications to protocol analysis and computational soundness. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2007.

[BMU08] M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Proceedings of 29th IEEE Symposium on Security and Privacy*. May 2008.

[BMV05] D. Basin, S. Mödersheim, and L. Viganò. A symbolic model checker for security protocols. *Journal of Information Security*, 4(3) :181–208, 2005.

[BN00] M. Bellare and C. Namprempre. Authenticated encryption : relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology (ASIACRYPT 2000)*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. 2000.

[BN05] J. Borgström and U. Nestmann. On bisimulations for the spi calculus. *Mathematical Structures in Computer Science*, 15(3) :487–552, 2005.

[Bon01] M. Bond. Attacks on cryptoprocessor transaction sets. In *Proceedings of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES'01)*, volume 2162 of *LNCS*, pages 220–234. Springer-Verlag, Paris (France), 2001.

[Bor01] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. of the 28th Int. Coll. Automata, Languages, and Programming (ICALP'01)*. Springer Verlag, July 2001.

[Bor05] J. Borgström. Static equivalence *is* harder than knowledge. In J. Baeten and I. Phillips, editors, *Proceedings of the 12th International Workshop on Expressiveness in Concurrency (EXPRESS'05)*, Electronic Notes in Theoretical Computer Science. Elsevier Science Publishers, San Francisco, CA, USA, August 2005.

[BP03] B. Blanchet and A. Podelski. Verification of cryptographic protocols : Tagging enforces termination. In A. Gordon, editor, *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *LNCS*. April 2003.

[BP04] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Science Foundations Workshop (CSFW'04)*, pages 204–218. 2004.

[BP05a] M. Backes and B. Pfitzmann. Limits of the cryptographic realization of dolev-yao-style xor and dolev-yao-style hash functions. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, Lecture Notes in Computer Science, pages 336–354. 2005.

[BP05b] M. Backes and B. Pfitzmann. Relating symbolic and computational secrecy. *Transactions on Dependable and Secure Computing*, 2(2) :109–123, 2005.

[BP05c] B. Blanchet and A. Podelski. Verification of cryptographic protocols : Tagging enforces termination. *Theoretical Computer Science*, 333(1-2) :67–90, March 2005. Special issue FoSSaCS'03.

[BPW03] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*. 2003.

[BPW07] M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12) :1685–1720, 2007.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In *Proceedings of the First Annual Conference on Computer and Communications Security, ACM*. 1993.

[BU08] M. Backes and D. Unruh. Computational soundness of symbolic zero-knowledge proofs against active attackers. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 255–269. IEEE Computer Society Press, Pittsburgh, PA, USA, June 2008.

[BV04] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *2nd ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN 2004)*. Washington DC, USA, October 2004.

[Can01] R. Canetti. Universal composable security : a new paradigm for cryptographic protocols. In *Symp. on Foundations of Computer Science*. 2001.

[CB03] R. Clayton and M. Bond. Experience using a low-cost FPGA design to crack DES keys. In *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded System (CHES'02)*, volume 2523 of *LNCS*, pages 579–592. Springer, Redwood Shores (CA, USA), 2003.

[CCA06] *CCA Basic Services Reference and Guide*, October 2006. Available online at `www.ibm.com/security/cryptocards/pdfs/bs327.pdf`.

[CCK+08] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Analyzing security protocols using time-bounded task-pioas. *Discrete Events Dynamic Systems*, 18(1), March 2008.

[CD99] K. Compton and S. Dexter. Proof techniques for cryptographic protocols. In *Proc. of the 26th International Colloquium on Automata, Languages, and Programming (ICALP'99)*. July 1999.

[CD05] H. Comon-Lundh and S. Delaune. The finite variant property : How to get rid of some algebraic properties. In J. Giesl, editor, *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307. Springer, Nara, Japan, April 2005. doi :10.1007/b135673.

[CDK09] Ş. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In *Proceedings of the 22nd International Conference on Automated Deduction (CADE'09)*. Montreal, Canada, 2009.

[CDL+99] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proc. of the 12th IEEE Computer Security Foundations Workshop (CSFW'99)*, pages 55–69. IEEE Computer Society Press, 1999.

[CH06] R. Canetti and J. Herzog. Universally composable symbolic analysis of cryptographic protocols. In *Theory of Cryptography Conference (TCC'06)*. 2006.

[Cio08] Ş. Ciobâcă. *Verification of anonymity properties in e-voting protocols*. Rapport de Master, Master Parisien de Recherche en Informatique, Paris, France, September 2008.

[CJ97] J. Clark and J. Jacob. A survey of authentication protocol literature, 1997. `http ://www.cs.york.ac.uk/~jac/ papers/drareviewps.ps`.

[CKR+03a] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Deciding the security of protocols with diffie-hellman exponentiation and product in exponents. In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*. 2003.

[CKR+03b] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Extending the dolev-yao intruder for analyzing an unbounded number of sessions. In *Proc. of the 17th International Workshop in Computer Science Logic (CSL'03)*. 2003.

[CKRT03] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with xor. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*. 2003.

[CKRT05] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. *Theoretical Computer Science*, 338(1-3) :247–274, June 2005.

[CL04] H. Comon-Lundh. Résolution de contraintes et recherche d'attaques pour un nombre borné de sessions, 2004. Available at `http://www.lsv.ens-cachan.fr/~comon/CRYPTO/bounded.ps`.

[CLN09] C. Cremers, P. Lafourcade, and P. Nadeau. Comparing state spaces in automatic security protocol analysis. In *Formal to practical Security*, number 5458 in Lecture Notes in Computer Science. 2009.

[CLS03] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 271–280. IEEE Computer Society, Los Alamitos, CA, 2003.

[Clu03a] J. Clulow. *The Design and Analysis of Cryptographic APIs for Security Devices.* Master's thesis, University of Natal, Durban, South Africa, 2003. Chapter 3.

[Clu03b] J. Clulow. On the security of pkcs#11. In *Proceedings of the 5th International Worshop on Cryptographic Hardware and Embedded Systems (CHES'03)*, volume 2779 of *LNCS*, pages 411–425. Springer-Verlag, Cologne (Germany), 2003.

[Cor06] R. Corin. *Analysis Models for Security Protocols.* Ph.D. thesis, University of Twente, 2006.

[CR03] R. Canetti and T. Rabin. Universal composition with joint state. Cryptology ePrint Archive, report 2002/47, Nov. 2003.

[CR08] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. *Information and Computation*, 206 :352–377, 2008.

[Cre08a] C. Cremers. The Scyther Tool : Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008. doi :10.1007/978-3-540-70545-1_38.

[Cre08b] C. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *CCS '08 : Proceedings of the 15th ACM conference on Computer and communications security*, pages 119–128. ACM, New York, NY, USA, 2008. ISBN 978-1-59593-810-7. doi :http://doi.acm.org/10.1145/1455770.1455787.

[DDM+05] A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic Polynomial-time Semantics for a Protocol Security Logic. In *Proc. of 32nd International Colloquium on Automata, Languages and Programming, ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2005. Lisboa, Portugal.

[DDMP05] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security (Special Issue of Selected Papers from CSFW-16)*, 13(3) :423–482, 2005.

[DDMW06] A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *Proceedings of 19th IEEE Computer Security Foundations Workshop*, pages 321–334. 2006.

[DEK83] D. Dolev, S. Even, and R. Karp. On the security of ping-pong protocols. In R. Rivest, A. Sherman, and D. Chaum, editors, *Proc. of CRYPTO 82*, pages 177–186. Plenum Press, 1983.

[Del06a] S. Delaune. Easy intruder deduction problems with homomorphisms. *Information Processing Letters*, 97(6) :213–218, March 2006.

[Del06b] S. Delaune. *Vérification des protocoles cryptographiques et propriétés algébriques.* Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, June 2006.

[DH76] W. Diffie and M. Helman. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6) :644–654, November 1976.

[DJ04] S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287. ACM Press, Washington, D.C., USA, October 2004.

[DKR06] S. Delaune, S. Kremer, and M. D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Computer Security Foundations Workshop (CSFW'06)*, pages 28–39. 2006.

[DKR07] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi-calculus. In V. Arvind and S. Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 133–145. Springer, New Delhi, India, December 2007. doi :10.1007/978-3-540-77050-3_11.

[DKR08] S. Delaune, S. Kremer, and M. D. Ryan. Composition of password-based protocols. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 239–251. IEEE Computer Society Press, Pittsburgh, PA, USA, June 2008. doi :10.1109/CSF.2008.6.

[DKR09a] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. *Journal of Computer Security*, 2009. To appear.

[DKR09b] S. Delaune, S. Kremer, and M. D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4) :435–487, 2009.

[DKS08] S. Delaune, S. Kremer, and G. Steel. Formal analysis of PKCS#11. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 331–344. IEEE Computer Society Press, Pittsburgh, PA, USA, June 2008.

[DLLT06] S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis in presence of a homomorphism operator and *exclusive or*. In M. Buglesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06) — Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 132–143. Springer, Venice, Italy, July 2006. doi :10.1007/11787006_12.

[DLLT08] S. Delaune, P. Lafourcade, D. Lugiez, and R. Treinen. Symbolic protocol analysis for monoidal equational theories. *Information and Computation*, 206(2-4) :312–351, February-April 2008. doi :10.1016/j.ic.2007.07.005.

[DLMS99] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*. 1999.

[DMP01] N. Durgin, J. Mitchell, and D. Pavlovic. A compositional logic for protocol correctness. In *14th IEEE Computer Security Foundations Workshop (CSFW'01)*. Cape Breton, Nova Scotia, June 2001.

[DMP03] N. Durgin, J. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4) :677–721, 2003.

[DSV03] L. Durante, R. Sisto, and A. Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Transactions on Software Engineering and Methodology*, 12(2) :222–284, 2003.

[DY81] D. Dolev and A. Yao. On the security of public key protocols. In *Proc. of the 22nd Symp. on Foundations of ComputerScience*, pages 350–357. IEEE Computer Society Press, 1981.

[ecc04] Council regulation (ec) no 2252/2004 : on standards for security features and biometrics in passports and travel documents issued by member states, December 2004. Available at `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2004:385:0001:0006:EN:PDF`.

[EGS86] S. Even, O. Goldreich, and A. Shamir. On the security of ping-pong protocols when implemented using the rsa. In *Advances in cryptology—CRYPTO 85*, volume 218 of *Lecture Notes in Computer Sciences*, pages 58–72. Santa Barbara, California, United States, 1986.

[Eis99] D. Eisenbud. *Commutative Algebra with a view toward Algebraic Geometry*. Springer, 1999.

[Eng85] J. Engelfriet. Determinacy implies (observation equivalence = trace equivalence). *Theoretical Computer Science*, 36 :21–25, 1985.

[FA01] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proc. of the 14th Computer Seurity Foundation Workshop (CSFW'01)*, pages 160–173. IEEE Computer Society Press, 2001.

[FA06] C. Fournet and P. Adao. Cryptographically sound implementations for communicating processes. In *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 83–94. Springer Verlag, 2006.

[FHF06] A. J. Feldman, J. A. Halderman, and E. W. Felten. Security analysis of the Diebold AccuVote-TS Voting Machine. http ://itpolicy.princeton.edu/voting/, 2006.

[Fis99] M. Fischlin. Pseudorandom function tribe ensembles based on one-way permutations : Improvements and applications. In *Advances in Cryptology - Eurocrypt 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 429–444. Springer-Verlag, 1999.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology - AUSACRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer-Verlag, 1992.

[Fro07] S. Froeschle. The insecurity problem : Tackling unbounded data. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF'07)*, pages 370–384. IEEE Computer Society, 2007.

[GD02] V. D. Gligor and P. Donescu. Fast encryption and authentication : Xcbc encryption and xecb authentication modes. In *Fast Software Encryption (FSE 2001)*, volume 2355 of *Lecture Notes in Computer Science*, pages 92–108. Springer, Yokohama, Japan, 2002.

[GGKL89] M. Gasser, A. Goldstein, C. Kaufmann, and B. Lampson. The digital distributed system security architecture. In *Proceedings of the 12th National Computer Security Conf., NIST/NCSC,*, pages 305–319. Baltimore, 1989.

[GJM99] J. A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Advances in Cryptology : Proc. of Crypto'99*, volume 1666 of *LNCS*, pages 449–466. Springer Verlag, 1999.

[GLRV04] J. Goubault-Larrecq, M. Roger, and K. N. Verma. Abstraction and resolution modulo AC : How to verify Diffie-Hellman-like protocols automatically. *Journal of Logic and Algebraic Programming*, 64(2) :219–251, 2004.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28 :270–299, April 1984.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2) :281–308, April 1988.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87 : Proc. of the 19th ACM Symposium on Theory of Computing*, pages 218–229. 1987. ISBN 0-89791-221-7. doi :http://doi.acm.org/10.1145/28395.28420.

[GT00] J. D. Guttman and F. J. Thayer. Protocol independence through disjoint encryption. In *Proc. 13th Computer Security Foundations Workshop (CSFW'00)*, pages 24–34. IEEE Comp. Soc. Press, 2000.

[GT01] J. Guttman and F. Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 2001.

[Gut04] J. D. Guttman. Authentication tests and disjoint encryption : a design method for security protocols. *Journal of Computer Security*, 12(3–4) :409–433, 2004.

[Gut09] J. D. Guttman. Cryptographic protocol composition via the authentication tests. In *Foundations of Software Science and Computation Structures (FOSSACS'09)*, Lecture Notes in Computer Science. March 2009.

[GvR08] F. D. Garcia and P. van Rossum. Sound and complete computational interpretation of symbolic hashes in the standard model. *Theoretical Computer Science*, 394 :112–133, 2008.

[GZA02] M. Guerrero Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002)*, pages 1–10. September 2002.

[Her03] J. Herzog. A computational interpretation of dolev-yao adversaries. In *Proceedings of the 3rd IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'03)*. 2003.

[Her05] J. Herzog. A computational interpretation of Dolev-Yao adversaries. *Theoretical Computer Science*, 340 :57–81, June 2005.

[HKT94] A. Herzberg, H. Krawczyk, and G. Tsudik. On travelling incognito. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*. 1994.

[HPJ03] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2003 ACM Workshop on Wireless Security (WiSe 2003)*, pages 30–40. ACM, San Diego, CA, 2003.

[HPJ05] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne : A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11 :21–38, 2005.

[Hut02] H. Huttel. Deciding framed bisimulation. In *4th International Workshop on Verification of Infinite State Systems INFINITY'02*, pages 1–20. 2002.

[IBM01] IBM Comment on "A Chosen Key Difference Attack on Control Vectors", January 2001. Available from `http ://www.cl.cam.ac.uk/~mkb23/research.html`.

[JLM05] R. Janvier, Y. Lakhnech, and L. Mazare. (de)compositions of cryptographic schemes and their applications to protocols. Cryptology ePrint Archive, Report 2005/020, 2005. `http://eprint.iacr.org/`.

[JLO97] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In *advances in cryptology, (CRYPTO-97)*, volume 1294 of *LNCS*, pages 150–164. 1997.

[JPVB08] M. Johansson, J. Parrow, B. Victor, and J. Bengtson. Extended pi-calculi. In *Proceedings of 33th International Colloquium of Automata, Languages and Programming (ICALP'08)*, volume 5126 of *Lecture Notes in Computer Science*, pages 87–98. Springer, 2008.

[KAG98] G. Karjoth, N. Asokan, and C. Gülcü. Protecting the computation results of free-roaming agents. In *Proc. the 2nd International Workshop on Mobile Agents (MA'98)*, volume 1477 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[KK05] D. Kähler and R. Küsters. Constraint Solving for Contract-Signing Protocols. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2005.

[KM07] S. Kremer and L. Mazaré. Adaptive soundness of static equivalence. In J. Biskup and J. Lopez, editors, *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 610–625. Springer, Dresden, Germany, September 2007. doi :10. 1007/978-3-540-74835-9_40.

[KM09] S. Kremer and L. Mazaré. Computationally sound analysis of protocols using bilinear pairings. *Journal of Computer Security*, 2009. To appear.

[KR02] S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In S. Schneider, editor, *Proc. of the 15th Computer Security Foundations Workshop (CSFW'02)*, pages 206–220. IEEE Computer Society Press, June 2002.

[KR05] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In M. Sagiv, editor, *Programming Languages and Systems – Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200. Springer, Edinburgh, U.K., April 2005.

[KSH08] Y. Kawamoto, H. Sakurada, and M. Hagiya. Computationally sound symbolic anonymity of a ring signature. In *Joint Workshop on Foundations of Computer Security, Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (FCS-ARSPA-WITS'08)*. Pittsburgh, USA, June 2008.

[KSW97] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proc. 5th International Workshop on Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 91–104. Springer, 1997.

[KT08] R. Küsters and M. Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computations. In *Computer Security Foundations (CSF'08)*. 2008.

[KY03] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proc. of Crypto'03*, pages 110–125. 2003.

[Lau02] P. Laud. Encryption cycles and two views of cryptography. In *Nordic Workshop on Secure IT Systems (NORDSEC'02)*. 2002.

[Lau04] P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proceedings of 2004 IEEE Symposium on Security and Privacy*, pages 71–85. Oakland, CA, May 2004.

[LBD+03] Lee, Boyd, Dawson, Kim, Yang, and Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *ICISC : International Conference on Information Security and Cryptology*. LNCS, 2003.

[LC04] P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In *Proc. 6th International Conference on Information Security and Cryptology (ICISC'03)*, volume 2971 of *Lecture Notes in Computer Science*, pages 55–66. Springer, 2004.

[LLT05] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for ac-like equational theories with homomorphisms. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, Nara, Japan, April 2005.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, march 1996.

[Low97] G. Lowe. Casper : A compiler for the analysis of security protocols. In *Proc. of 10th Computer Security Foundations Workshop (CSFW'97)*. IEEE Computer Society Press, Rockport, Massachusetts, USA, 1997. Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.

[LR92] D. Longley and S. Rigby. An automatic search for security flaws in key management schemes. *Computers and Security*, 11(1) :75–89, March 1992.

[LR97] G. Lowe and A. Roscoe. Using CSP to detect errors in the TMN protocol. In *IEEE transactions on Software Engineering*, volume 23, pages 659–669. 1997.

[Maz07] L. Mazaré. Computationally sound analysis of protocols using bilinear pairings. In *Proc. 7th International Workshop on Issues in the Theory of Security (WITS'07)*, pages 6–21. 2007.

[Mer83] M. J. Merritt. *Cryptographic Protocols*. Ph.D. thesis, Georgia Institute of Technology, February 1983.

[Mil99] R. Milner. *Communicating and Mobile Systems : the Pi-Calculus*. Cambridge University Press, 1999.

[Mil03] J. Millen. On the freedom of decryption. *Information Processing Letters*, 2003.

[MS01] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of the 8th ACM Conference on Computer and Communications Security (CCS'01)*. 2001.

[MS03] J. Millen and V. Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation. In *Proc. of the 16th IEE Computer Seurity Foundation Workshop (CSFW'03)*. 2003.

[MW04a] D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway language of encrypted expressions. *Journal of Computer Security*, 2004.

[MW04b] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer-Verlag, Cambridge, MA, USA, February 2004.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12) :993–999, 1978.

[Nut90] W. Nutt. Unification in monoidal theories. In *Proc. 10th Int. Conference on Automated Deduction, (CADE'90)*, volume 449 of *LNCS*, pages 618–632. Springer, Kaiserslautern (Germany), 1990.

[PP04] D. H. Phan and D. Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *Proc. Selected Areas in Cryptography (SAC'04)*, volume 3357 of *Lecture Notes in Computer Science*, pages 185–200. 2004.

[PR99] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100. New Orleans, LA, February 1999.

[PS00] A. Perrig and D. Song. Looking for diamonds in the desert. In *Proc. of the 13th Computer Security Foundations Workshop (CSFW'00)*, pages 64–76. IEEE Computer Society Press, 2000.

[RBB03] P. Rogaway, M. Bellare, and J. Black. Ocb : A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3) :365 – 403, 2003.

[RDDM07] A. Roy, A. Datta, A. Derek, and J. C. Mitchell. Inductive proofs of computational secrecy. In *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2007.

[RDM82] N. L. R. Demillo and M. Merritt. Cryptographic protocols. In *Proc. of the 14^{th} ACM SIGACT Symposium on Theory of Computing*. ACM, May 1982.

[RH07] M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1) :39–68, 2007.

[Rot01] V. Roth. On the robustness of some cryptographic protocols for mobile agent protection. In *Mobile Agents*, volume 2240 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[RS03] R. Ramanujam and S.P.Suresh. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*. Mumbai, 2003.

[RSA04] RSA Security Inc., v2.20. *PKCS #11 : Cryptographic Token Interface Standard.*, June 2004.

[RSG+00] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and A. Roscoe. *The modelling and analysis of security protocols : the CSP approach.* Addison-Wesley, 2000.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Computer Society Press, 2001.

[RT03] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299 :451–475, 2003.

[SBB+06] C. Sprenger, M. Backes, D. Basin, B. Pfitzmann, and M. Waidner. Cryptographically sound theorem proving. In *Proc. 19th IEEE Computer Science Foundations Workshop (CSFW'06)*, pages 153–166. July 2006.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986.

[Sch96a] S. Schneider. Security properties and CSP. In *Proc. of the Symposium on Security and Privacy*, pages 174–187. IEEE Computer Society Press, 1996.

[Sch96b] B. Schneier. *Applied Cryptography Second Edition : protocols, algorithms, and source code in C.* J. Wiley & Sons, Inc., 1996.

[Sch97] S. Schneider. Verifying authentication protocols with CSP. In *Proc. of the 10th Computer Security Foundations Workshop (CSFW'97).* IEEE Computer Society Press, 1997.

[SDL+02] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10 th IEEE International Conference on Network Protocols (ICNP'02).* 2002.

[SH02] V. Shmatikov and D. Hughes. Defining Anonymity and Privacy (extended abstract). In *Proc. of the Workshop on Issues in the Theory of Security (WITS '02).* 2002.

[Shm04] V. Shmatikov. Decidable analysis of cryptographic protocols with products and modular exponentiation. In *Proc. 13th European Symposium On Programming (ESOP'04)*, volume 2986 of *LNCS*, pages 355–369. Springer-Verlag, Barcelona (Spain), 2004.

[SKY05] R. Song, L. Korba, and G. Yee. Anondsr : Efficient anonymous dynamic source routing for mobile ad-hoc networks. In *Proceedings of the 2005 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005)*, pages 32–42. Alexandria, Virginie, 2005.

[SM00] V. Shmatikov and J. Mitchell. Analysis of abuse-free contract signing. In *Financial Cryptography '00*, volume 1962 of *LNCS*. Springer Verlag, February 2000.

[SMA95] D. Samfat, R. Molva, and N. Asokan. Untraceability in mobile networks. In *Mobile Computing and Networking*, pages 26–36. 1995.

[SS96] S. Schneider and A. Sidiropoulos. CSP and anonymity. In *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, LNCS. Springer Verlag, 1996.

[SSBC09] P. Schaller, B. Schmidt, D. Basin, and S. Capkun. Modeling and verifying physical properties of security protocols for wireless networks. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF'09)*, pages 109–123. 2009.

[SV05] H. Seidl and K. N. Verma. Flat and one-variable clauses : Complexity of verifying cryptographic protocols with single blind copying. In *Logic for Programming and Automated Reasoning (LPAR'04)*, LNCS, pages 79–94. Springer-Verlag, 2005.

[SV08] H. Seidl and K. N. Verma. Flat and one-variable clauses : Complexity of verifying cryptographic protocols with single blind copying. *ACM Transactions on Computational Logic*, 2008.

[THG99] J. Thayer, J. Herzog, and J. Guttman. Strand spaces : proving security protocols correct. *IEEE Journal of Computer Security*, 7 :191–230, 1999.

[Tur03] M. Turuani. *Sécurité des protocoles cryptographiques : décidabilité et complexité.* Ph.D. thesis, Université Henri Poincaré, Nancy, France, décembre 2003.

[Tur06] M. Turuani. The CL-Atse Protocol Analyser. In *Term Rewriting and Applications - Proc. of RTA*, volume 4098 of *Lecture Notes in Computer Science*, pages 277–286. Seattle, WA, USA, 2006. ISBN 3-540-36834-5.

[Tur09] A. Turrini. *Hierarchical and Compositional Verification of Cryptographic Protocols.* Ph.D. thesis, University of Verona, 2009.

[Ver03] K. N. Verma. Two-way equational tree automata for AC-like theories : Decidability and closure properties. In *Proc. 14th International Conference on Rewriting Techniques and Applications (RTA'03)*, volume 2706 of *LNCS*, pages 180–196. Springer-Verlag, Valencia (Spain), 2003.

[VSS05] K. N. Verma, H. Seidl, and T. Schwentick. On the complexity of equational horn clauses. In *Proc. of the 22th International Conference on Automated Deduction (CADE 2005)*, Lecture Notes in Computer Science, pages 337–352. Springer-Verlag, 2005.

[War03] B. Warinschi. A computational analysis of the Needham-Schroeder protocol. In *Proc. 16th IEEE Computer Science Foundations Workshop (CSFW'03)*, pages 248–262. 2003.

[War05] B. Warinschi. A computational analysis of the Needham-Schroeder protocol. *Journal of Computer Security*, 13 :565–591, 2005.

[Wei99] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In H. Ganzinger, editor, *Proc. of the 16th International Conference on Automated Deduction, CADE'99*, volume 1632 of *LNCS*, pages 378–382. 1999.

Bibliographie

Revues internationales

[CCLZ10] V. Cortier, H. Comon-Lundh, and E. Zalinescu. Deciding security properties for cryptographic protocols. Application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 2010. To appear.

[BCK09] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. *Information and Computation*, 207(4) :496–520, April 2009.

[CD09c] V. Cortier and S. Delaune. Safely composing security protocols. *Formal Methods in System Design*, 34(1) :1–36, 2009.

[CRZ07] V. Cortier, M. Rusinowitch, and E. Zalinescu. Relating two standard notions of secrecy. *Logical Methods in Computer Science*, 3(3), July 2007.

[AC06] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2) :2–32, November 2006.

[CGLN06] V. Cortier, X. Goaoc, M. Lee, and H.-S. Na. A note on maximally repeated sub-patterns of a point set. *Discrete Mathematics*, 306(16) :1965–1968, August 2006.

[CDL06] V. Cortier, S. Delaune, and P. Lafourcade. A Survey of Algebraic Properties Used in Cryptographic Protocols. *Journal of Computer Security*, 14(1/2006), 2006.

[CLC05] H. Comon-Lundh and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science*, 331(1) :143–214, February 2005.

[CLC04] H. Comon-Lundh and V. Cortier. Security properties : two agents are sufficient. *Science of Computer Programming*, 50(1-3) :51–71, March 2004.

[Cor02a] V. Cortier. About the decision of reachability for register machines. *Theoretical Informatics and Applications*, 36(4) :341–358, 2002.

Revues nationales

[Cor05] V. Cortier. Vérifier les protocoles cryptographiques. *Technique et Science Informatique, Hermes Science*, 24(1) :115–140, 2005.

Édition de livres

[CKOS09] V. Cortier, C. Kirchner, M. Okada, and H. Sakurada, editors. *Formal to practical Security*, volume 5458 of *Lecture Notes in Computer Science*. Springer, springer edition, 2009.

Chapitres de livres

[Cor06b] V. Cortier. *Cryptographie et codes secrets*, chapter Les protocoles cryptographiques, pages 106–113. Bibliothèque Tangente, POLE edition, 2006.

[Cor06e] V. Cortier. *Sur les chemins de la découverte*, chapter Sécuriser les réseaux, les protocoles cryptographiques, pages 107–118. Presses Universitaires de France, January 2006.

Conférences internationales

[CS09a] V. Cortier and G. Steel. A generic security API for symmetric key management on cryptographic devices. In *Proceedings of the 14th European Symposium On Research In Computer Security (ESORICS'09)*, volume 5789 of *Lecture Notes in Coputer Science*, pages 605–620. Springer, St Malo, France, September 2009.

[CD09a] V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF'09)*, pages 266–276. IEEE Computer Society Press, Port Jefferson, NY, USA, July 2009.

[BCD09a] M. Baudet, V. Cortier, and S. Delaune. YAPA : A generic tool for computing intruder knowledge. In *20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science*, pages 148–163. Springer, Brasília, Brazil, June 2009.

[Cor09b] V. Cortier. Verification of security protocols *(invited tutorial)*. In *10th Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'09)*, volume 5403 of *Lecture Notes in Computer Science*, pages 5–13. Springer, Savanah, USA, January 2009.

[CLC08a] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. ACM Press, Alexandria, Virginia, USA, October 2008.

[CDD07a] V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In V. Arvind and S. Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 352–363. Springer, New Delhi, India, December 2007.

[CWZ07b] V. Cortier, B. Warinschi, and E. Zalinescu. Synthetizing secure protocols. In *Proceedings of the 12th European Symposium On Research In Computer Secu-*

rity (ESORICS'07), volume 4734, pages 406–421. Springer, Dresden, Germany, September 2007.

[CKW07a] V. Cortier, R. Küsters, and B. Warinschi. A cryptographic model for branching time security properties – the case of contract signing protocols. In *Proceedings of the 12th European Symposium On Research In Computer Security (ESORICS'07)*, volume 4734, pages 422–437. Springer, Dresden, Germany, September 2007.

[CD07a] V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In *Proc. of the 14th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 196–210. Springer, Yerevan, Armenia, October 2007.

[ACD07a] M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. In *Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 103–117. Springer, Liverpool, UK, September 2007.

[CDS07a] V. Cortier, S. Delaune, and G. Steel. A formal theory of key conjuring. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 79–93. IEEE Computer Society Press, Venice, Italy, July 2007.

[CKS07] V. Cortier, G. Keighren, and G. Steel. Automatic analysis of the security of xor-based key management schemes. In *13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07)*, volume 4424 of *Lecture Notes in Computer Science*, pages 538–552. Springer, Braga, Portugal, March 2007.

[CKKW06] V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In N. Garg and S. Arun-Kumar, editors, *Proceedings of the 26th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 176–187. Springer, Kolkata, India, December 2006.

[CZ06] V. Cortier and E. Zalinescu. Deciding key cycles for security protocols. In *Proc. of the 13th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 317–331. Springer, Phnom Penh, Cambodia, November 2006.

[CRZ06] V. Cortier, M. Rusinowitch, and E. Zalinescu. Relating two standard notions of secrecy. In Z. Esik, editor, *Proceedings of 20th Int. Conference on Computer Science Logic (CSL'06)*, volume 4207 of *Lecture Notes in Computer Science*, pages 303–318. Springer, Szeged, Hungary, September 2006.

[BCK05] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663. Springer, Lisboa, Portugal, July 2005.

[CRZ05] V. Cortier, M. Rusinowitch, and E. Zalinescu. A resolution strategy for verifying cryptographic protocols with CBC encryption and blind signatures. In *Proc. of*

the 7th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'05), pages 12–22. ACM press, Lisboa, Portugal, July 2005.

[AC05] M. Abadi and V. Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 62–76. IEEE Comp. Soc. Press, Aix-en-Provence, France, June 2005.

[CW05] V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171. Springer, Edinburgh, U.K, April 2005.

[AC04a] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. 31st Int. Coll. Automata, Languages, and Programming (ICALP'2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 46–58. Springer, Turku, Finland, July 2004.

[CLC03a] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 148–164. Springer-Verlag, 2003.

[CLC03c] H. Comon-Lundh and V. Cortier. Security properties : two agents are sufficient. In *Proc. of the 12th European Symposium On Programming (ESOP'03)*, volume 2618 of *LNCS*, pages 99–113. Springer Verlag, April 2003.

[CCM01] H. Comon, V. Cortier, and J. Mitchell. Tree automata with one memory, set constraints and ping-pong protocols. In *Proc. of the 28th Int. Coll. Automata, Languages, and Programming (ICALP'01)*, volume 2076, pages 682–693. Springer Verlag, July 2001.

[CMR01] V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 97–108. IEEE Computer Society Press, 2001.

[CC00] H. Comon and V. Cortier. Flatness is not a weakness. In *Proc. of the 14th International Workshop Computer Science Logic (CSL'2000)*, volume 1862, pages 262–276. Springer Verlag, 2000.

[CGJV99] V. Cortier, H. Ganzinger, F. Jacquemard, and M. Veanes. Decidable fragments of simultaneous rigid reachability. In *Proc. of the 26th Int. Coll. Automata, Languages, and Programming (ICALP'99)*, volume 1644, pages 250–260. Springer Verlag, July 1999.

Workshops

[CCL08] V. Cortier and H. Comon-Lundh. Computational soundness of observational equivalence. In *4th Workshop on Formal and Computational Cryptography (FCC 2008)*. Pittsburgh, United States, June 2008.

[CD07b] V. Cortier and S. Delaune. Deciding knowledge in security protocols for monoidal equational theories. In P. Degano, R. Küsters, L. Viganò, and S. Zdancewic,

editors, *Proceedings of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'07)*, pages 63–80. Wrocław, Poland, July 2007.

[CKW07b] V. Cortier, R. Küsters, and B. Warinschi. A cryptographic model for branching time security properties – the case of contract signing protocols. In *3rd Workshop on Formal and Computational Cryptography (FCC 2007)*. Venice, Italy, 2007.

[CZ07] V. Cortier and E. Zalinescu. Deciding key cycles for security protocols. In *3rd Workshop on Formal and Computational Cryptography (FCC 2007)*. Venice, Italy, 2007.

[CS06] V. Cortier and G. Steel. On the decidability of a class of xor-based key-management APIs. In *Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'06)*. Seattle, Washington, August 2006.

[CHW07] V. Cortier, H. Hördegen, and B. Warinschi. Explicit randomness is not necessary when modeling probabilistic encryption. In *Workshop on Information and Computer Security (ICS 2006)*, volume 186 of *Electronic Notes Theoretical Computer Science*, pages 49–65. Timisoara, Romania, September 2007.

Vulgarisation scientifique

[Cor06a] V. Cortier. Ces protocoles qui nous protègent. *Tangente*, Hors-série 26 :42–44, July 2006.

[Cor06c] V. Cortier. Divers protocoles couramment utilisés en informatique. *Tangente*, Hors-série 26 :45, July 2006.

[Cor06d] V. Cortier. Protocoles cryptographiques : analyse par méthodes formelles. In *Techniques de l'ingénieur*, volume dossier AF176, chapter Bases documentaires "Mathématiques pour l'ingénieur". April 2006.

Rapports de recherche

[CKW09] V. Cortier, S. Kremer, and B. Warinschi. A Survey of Symbolic Methods in Computational Analysis of Cryptographic Systems. Research Report RR-6912, INRIA, 2009.

[BBRC09] M. Berrima, N. Ben Rajeb, and V. Cortier. Deciding knowledge in security protocols under some e-voting theories. Research Report RR-6903, INRIA, April 2009.

[CS09b] V. Cortier and G. Steel. Synthesising secure APIs. Research Report RR-6882, INRIA, March 2009.

[BCD09b] M. Baudet, V. Cortier, and S. Delaune. YAPA : A generic tool for computing intruder knowledge. Research Report LSV-09-03, Laboratoire Spécification et Vérification, ENS Cachan, France, February 2009. 26 pages.

[CD09b] V. Cortier and S. Delaune. A method for proving observational equivalence. Research Report LSV-09-04, Laboratoire Spécification et Vérification, ENS Cachan, France, February 2009. 21 pages.

[CLC08b] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. Research Report 6508, INRIA, 04 2008.

[CDD07b] V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. Research Report 6234, INRIA, June 2007.

[CWZ07a] V. Cortier, B. Warinschi, and E. Zalinescu. Synthesizing secure protocols. Research Report 6166, INRIA, April 2007.

[ACD07b] M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. Research Report 6118, INRIA, February 2007.

[CDS07b] V. Cortier, S. Delaune, and G. Steel. A formal theory of key conjuring. Research Report RR-6134, INRIA, February 2007.

[CW04] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. Research Report RR-5341, INRIA, October 2004.

[AC04b] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. Technical Report RR-5169, INRIA, April 2004.

[CDL04a] V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. Technical Report LSV-04-15, Laboratoire Spécification and Vérification, ENS de Cachan, 2004.

[CLC03b] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. Technical Report LSV-03-3, Laboratoire Specification and Verification, ENS de Cachan, January 2003. 30 pages.

[CLC02] H. Comon-Lundh and V. Cortier. Security properties : two agents are sufficient. In *Research Report LSV-02-10*. August 2002.

[Cor02c] V. Cortier. Observational equivalence and trace equivalence in an extension of spi-calculus. application to cryptographic protocols analysis. In *Research Report LSV-02-3*. March 2002.

[CC01] H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. In *Research Report LSV-01-13*. december 2001.

Rapports de contrats

[Cor03a] V. Cortier. *A guide for Securify*. RNTL EVA project, rapport numéro 13 edition, 2003.

[CDL04b] V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. Technical Report 2, projet RNTL PROUVÉ, June 2004.

[Cor02b] V. Cortier. Outil de vérification securify. Technical Report 7, projet RNTL EVA, May 2002.

Mémoires

[Cor03b] V. Cortier. *Vérification automatique des protocoles cryptographiques.* Ph.D. thesis, École Normale Supérieure de Cachan, Cachan, France, March 2003.

[Cor99] V. Cortier. *Vérification de Systèmes à Compteurs.* Master's thesis, Université Paris 7, 1999.

Listes de co-auteurs

Mathilde Arnaud
Martín Abadi
Mathieu Baudet
Narjes Ben Rajeb
Mouhebeddine Berrima
Hubert Comon-Lundh
Jérémie Delaitre
Stéphanie Delaune
Harald Ganzinger
Xavier Goaoc
Heinrich Hördegen
Florent Jacquemard
Gavin Keighren
Steve Kremer
Ralf Küsters
Pascal Lafourcade
Mira Lee
Jon Millen
John Mitchell
Hyeon-Suk Na
Harald Ruess
Michael Rusinowitch
Graham Steel
Margus Veanes
Bogdan Warinschi
Eugen Zalinescu